

# Minimizing Average Startup Delay of Periodic VOD Broadcasting Schedules

Yi Sun, *Nonmember* and Tsunehiko Kameda\*, *Nonmember*

**Abstract**—Consider a video consisting of a sequence of fixed-size pages to be transmitted over  $c$  broadcast channels such that a page fits in a slot of a channel. In *Shifted Harmonic Windows scheduling*, page  $i$  must be scheduled at least once in any consecutive  $i+m-1$  slots for some integer  $m \geq 1$ , to ensure continuous display of successive pages by the viewer, starting at the  $m^{\text{th}}$  slot boundary after tuning in. As  $m$  is increased, the maximum startup delay for Shifted Harmonic Windows schedules approaches the well-known general lower bound on the maximum startup delay,  $1/(e^c - 1)$ . We first derive a tight lower bound, which is lower than  $1/(e^c - 1)$ , on the average startup delay of periodic schedules. This settles one of the long-standing open problems about the optimal average performance of periodic broadcasting schedules. We then show that the average startup delay of Shifted Harmonic Windows schedules cannot approach this lower bound. Using a new framework called *Harmonic Block Windows scheduling* with a parameter  $b (\geq 1)$ , we present a scheduling algorithm which generates schedules whose average startup delays approach the lower bound as  $b$  is increased.

**EDICS: 5-STRM Index Terms**—Multimedia, Video-on-demand, broadcasting, scheduling, channels, bandwidth optimization

## I. INTRODUCTION

In all video-on-demand (VoD) broadcasting schemes proposed in recent years, a video is divided into a sequence of segments that are broadcast in different channels or in one or more shared channels. We define two policies for downloading and display that are commonly used. Under the *fixed-delay (FD) policy*, the user initially waits for a fixed amount of time before viewing starts. Under the *fixed start points (FSP) policy*, on the other hand, the user starts viewing the video at one of the prespecified points in time. We shall call a scheme based on the FD (FSP) policy an *FD (FSP) scheme*, and a schedule of an FD (FSP) scheme an *FD (FSP) schedule*. Numerous FD and FSP schemes have been proposed recently. A non-exhaustive list of FD schemes includes Greedy Equal Bandwidth Broadcasting [1], Polyharmonic Broadcasting [2], and Fixed-Delay Pagoda Broadcasting [3], while Pyramid Broadcasting [4], Harmonic Broadcasting [5], Skyscraper Broadcasting [6], Pagoda Broadcasting [7], Quasi-Harmonic Broadcasting [2], etc., are FSP schemes.

Suppose a video is divided into a sequence of fixed-size pages such that a page fits in a *slot* of a channel. Hollmann

and Holzschcherer [8] first proposed the FSP policy that works as follows. A user who tunes in at an arbitrary time waits until the beginning of the next slot boundary before s/he starts to download, and starts viewing at the  $m^{\text{th}}$  block boundary, where  $m \geq 1$  is an integer parameter. *Harmonic Broadcasting* [5] is a special case where  $m = 1$ . Note that if there are  $c$  channels, pages can be downloaded from up to  $c$  slots concurrently.

In our model of the FSP schemes used in this paper each channel is divided into a sequence of slots and a fixed number,  $b$ , of slots forms a *slot block*. We introduce a sub-page unit of data called a *fragment* that fits in a slot such that  $b$  fragments constitute a (normal) page. Therefore, a page fits in a slot block. The parameter  $m$  mentioned in the previous paragraph now refers to the number of blocks. Downloading starts only at a block boundary, and viewing starts at the same or later block boundary. Harmonic Broadcasting is a special case where  $m = b = 1$ . It might appear that we wouldn't gain anything by using  $b > 1$ , since the same effect could be achieved by increasing  $m$ . This is true for the maximum startup delay, but it turns out that the average startup delay is minimized when  $m = 1$  and  $b > 1$ , as we shall show.

Two performance metrics for VoD broadcasting schemes that are generally considered most important are the server bandwidth and the startup delay, and there is a trade-off between the two. Since the FSP policy implies variable startup delay, we consider the *average* startup delay for Poisson arrivals as our main performance metric. Note that for the FD schemes, the average and maximum delays are the same. Moreover, the (tight) lower bound on the *maximum* startup delays for the FSP schedules is the same as the (tight) lower bound on the startup delays for the FD schedules. It turns out, however, that the (tight) lower bound on the startup delays for the FD schedules is from 12% to 16% more than the average startup delay that is actually achievable by FSP schedules for the cases where one to eight channels are used. (See Table III.) There is a recent work on average startup delays that treat simple cases [9].

The rest of this paper is organized as follows. In Section II, we introduce our model of the FSP schemes to be used in the subsequent sections and review related work. Section III presents our main result, which is the (tight) lower bound on the average startup delay for the FSP schemes. Section IV presents a new scheduling algorithm that generates schedules whose average startup delays are asymptotically optimal. Section V compares the lower bounds on the average startup delays for the FD and FSP schedules, proving that there are practical FSP schemes that can outperform (in the average startup delay) any FD scheme that uses the same bandwidth.

Manuscript received December 25, 2006. This work was supported in part by the Natural Science and Engineering Research Council of Canada.

Y. Sun is with the College of Information Sciences and Engineering, Graduate University of the Chinese Academy of Sciences, Beijing, People's Republic of China, Email: sunyi@gucas.ac.cn, Phone: +86(10)806-16059, and T. Kameda\* is with the School of Computing Science, Simon Fraser University, Burnaby, B.C. V5A 1S6 Canada, Email: tiko@cs.sfu.ca, Phone: +1(778)883-9577, Fax: +1(604)291-3045

Finally Section VI concludes the paper with some remarks.

## II. PRELIMINARIES

### A. Models

Throughout this paper we assume that the given video is encoded at a *constant bit rate* of  $r$  (bits/sec), called the *display rate*. We use  $c$  channels of bandwidth  $r$  (bits/sec) each, where  $c \geq 1$ . Suppose that each channel is slotted and  $b$  consecutive *slots* constitute a *slot block*, or just a *block* for short. For now we assume that the length of each slot is fixed, but later we vary it to reflect the size of a “fragment” of data. The blocks on all channels are synchronized in the sense that they start and end at the same time, as shown in Fig. 1, where a *slice* consists of  $c$  concurrent slots, one from each channel. From now on, a block may sometimes refer to a collection of  $c$  blocks, one block from each channel, i.e.,  $b$  slices.

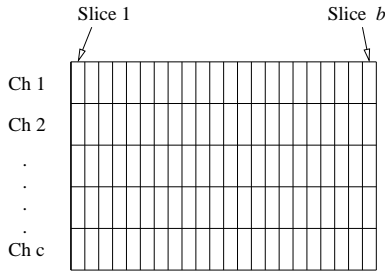


Fig. 1. A block consisting of  $b$  slices or  $bc$  slots.

Let us use a unit of data, called a *fragment*, of size  $\delta$  (bits) that fits exactly in a slot. Each page consists of  $b$  fragments, thus the page (normal) size is given by  $B = b\delta$ . Let  $P_i = \{f_{ij} \mid 1 \leq j \leq b\}$  be the set of fragments of page  $i$  ( $1 \leq i \leq n$ ).  $P_{n+1}$  contains less than  $b$  fragments. Scheduling consists in assigning the fragments of the video to slots of channel blocks in such a way that if the viewer starts downloading at any block boundary, it is guaranteed that all fragments of the video will be available for seamless display. Under the FSP policy, a user who tunes in at an arbitrary time waits until the beginning of the next block boundary before s/he starts to download, and starts viewing at the  $m^{\text{th}}$  block boundary.

For any given  $i$ , starting from some time origin, we can imagine that the slot blocks are partitioned into *block windows* of size  $i$ , each consisting of  $i$  consecutive blocks.

Let us consider the simple case where block size  $b = 1$  (and  $P_{n+1} = \emptyset$ ) for the time being, so that an entire page fits in a slot. We call this special case *Shifted Harmonic Windows scheduling*, although it was introduced in [10] as *Harmonic Windows scheduling with shifting technique*. Pâris proved the following lemma in the context of a FD scheme, but it is valid also for the FSP schemes [10].

*Lemma 1:* [3] In the case  $b = 1$ , in order to be able to display a video without interruptions, it is necessary and sufficient that, for all  $i$  ( $1 \leq i \leq n$ ), page  $i$  appear at least once in any block window of size  $i + m - 1$ . ■

The above lemma implies that page  $i$  requires bandwidth at least  $r/(i + m - 1)$ . Let  $H_n$  denote the  $n^{\text{th}}$  *harmonic number*

defined by  $H_n = \sum_{i=1}^n 1/i$ . Then the total bandwidth required by all the  $n$  pages is at least

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

In [11] it is shown that

$$H_{n+m-1} - H_{m-1} = \ln(1 + n/m) + \frac{1}{2m(1 + m/n)} + o(m^{-2})$$

Let us consider the extreme case where each page consists of just one bit (conceptually, it can be even smaller than a bit), and assume that it takes  $w$  (sec) to send  $m$  bits on a channel. The maximum and average startup delays are thus given by  $w_{max} = (m/n)D$  and  $w_{aver} = ((m-0.5)/n)D$ , respectively, assuming Poisson arrivals, where  $D$  (sec) is the duration of the video. If we let  $n \rightarrow \infty$  while keeping  $w_{max}$  fixed, then from (1) and (2) we get the following lower bound on the required number of channels  $c$ :

$$c \geq \ln(1 + 1/w_{max}^*), \quad (3)$$

where  $w_{max}^* = w_{max}/D$  is the normalized maximum startup delay. Eq. (3) can be rewritten as the lower bound on the normalized startup delay as follows:

$$w_{max}^* \geq \frac{1}{e^c - 1}. \quad (4)$$

Note that the above extreme case ( $n, m \rightarrow \infty$ ) corresponds to the FD policy.<sup>1</sup> In this extreme case of Shifted Harmonic Windows scheduling, the user waits for just at most one slot time,  $D/n \approx 0$ , before starting to download, i.e., downloading starts immediately.

In an FD scheme a “page” may consist of just one bit. However, it is impractical to broadcast each such “page”  $i$  with its own period of  $i + m - 1$  (bits). Therefore, in a practical FD schemes such as Greedy Equal Bandwidth Broadcasting [1] and Polyharmonic Broadcasting [2], a small number of segments consisting of different numbers of “pages” (=bits) are used, and all “pages” in a segment are broadcast with the same period.

As for the average startup delay, note that we have  $w_{aver} = ((m-0.5)/n)D = ((m-0.5)/m)w_{max}$ . Due to the factor  $(m-0.5)/m$ , it turns out that  $m = 1$  minimizes it.

It takes  $T_B = B/r$  (sec) to download the data transmitted in a block of a channel, where  $B$  is the number of bits contained in each page (except the last page). The block boundaries are the *fixed start points* where a viewer can start downloading and/or viewing. The average startup delay is given by  $(m-0.5)T_B = (m-0.5)B/r$ , since the viewer who tunes in at an arbitrary time needs to wait half the block time on average before downloading can start, assuming Poisson arrivals. Similarly, the maximum startup delay is given by  $mT_B = mB/r$ . As  $m \rightarrow \infty$ , the maximum and average startup delays of the FSP schemes converge to the same value, as we saw above. As the performance metric, we use the normalized average startup delay  $T_w^* = (m-0.5)T_B/D$ .

<sup>1</sup>In the FD schemes the bandwidth of each channel can be smaller than the display rate, since every segment is first buffered before display.

Let  $F_k$  denote a  $c$ -tuple of fragments, one from each channel. A schedule for  $c$  channels is a semi-infinite sequence of such  $c$ -tuples,  $S = F_1 F_2 \dots$ . We also consider that a schedule consists of sequence of blocks, each containing  $b$   $c$ -tuples,  $S = [F_1 \dots F_b][F_{b+1} \dots F_{2b}] \dots$ . A fragment is said to have a *block period*  $i$  in schedule  $S$ , if it appears once in every  $i$  consecutive blocks of  $S$ . In the rest of the paper (except in the next subsection), we consider schedules in which each fragment appears with a fixed block period.

### B. Related work

Pâris proposed Fixed-Delay Pagoda Broadcasting as an FD scheme [3], but its schedules can be used under the FSP policy as well. Used under the FSP policy, they become the *Shifted Harmonic Windows* schedules, which were originally conceived in [8]. Bar-Noy et al. [10] considered it as a *shifting* technique for improving the performance of HW schedules and used *round-robin trees* [12] to find schedules.<sup>2</sup> But they used an exhaustive search for the optimal value of  $\Delta$  (the root degree of the 2-level round-robin tree), trying  $O(m)$  possible cases. They gave an upper bound on the maximum startup delay that approaches the lower bound  $1/(e^c - 1)$  asymptotically as  $m \rightarrow \infty$ . The search space for  $\Delta$  was reduced to  $O(\sqrt{m})$  in [13].

Quasi-Harmonic Broadcasting (QHB) [2] is a bandwidth-efficient FSP scheme. In fact, its maximum bandwidth requirement approaches the lower bound (1) for Harmonic Windows scheduling ( $m = 1$ ), i.e.,  $rH_n$ , as one of its parameters approaches  $\infty$ . Its disadvantages are that each page is divided into a different number of fragments (the fragment size depends on the page) and that each channel requires a different amount of bandwidth. All this makes QHB difficult to implement. We could define the “shifted” QHB by slightly modifying QHB. However, that doesn’t make it any easier to implement.

### C. Tree representation of a schedule

Bar-Noy et al. introduced the *round robin tree* (*RR-tree*, for short), which is a rooted tree useful for representing a cyclic schedule for a channel [12]. Let  $\mathcal{L}_T$  denote the set of all leaves of RR-tree  $T$ . We label each leaf in  $\mathcal{L}_T$  by a fragment to represent a schedule, obtained by the following rule:

#### RR-tree schedule

- 1) Initially the root of  $T$  gets a turn.
- 2) When a non-leaf node gets a turn, it passes the turn to its “next” child node. (The leftmost child node gets a turn first and the order “next” means the next sibling to the right, wrapping around back to the leftmost child node.)
- 3) When a leaf gets a turn, its associated fragment is scheduled and the turn goes to the root. ■

In this paper we use only *RR<sup>2</sup>-trees*, i.e., 2-level RR-trees, whose root degree equals the block size  $b$ . For a leaf node  $v$  of an RR<sup>2</sup>-tree  $T$ , the product of the degrees of all its ancestor

nodes in  $T$  is the period of  $v$  in the schedule generated by the above rule [12]. Therefore, the degree of a subtree is the block period of the fragments assigned to the subtree. In this paper we consider only *RR<sup>2</sup>-schedules* that are represented by RR<sup>2</sup>-trees. The *offset*,  $o_T(v)$ , of leaf node  $v$  in an RR<sup>2</sup>-tree is the number of subtrees that are to the left of the subtree to which  $v$  belongs.

TABLE I  
AN EXAMPLE OF A BLOCK-UNIFORM SCHEDULE (ROWS 3 AND 4 ARE CONTINUATIONS OF ROWS 1 AND 2, RESPECTIVELY).

Ch1	$f_{11}$	$f_{12}$	$f_{13}$	$f_{11}$	$f_{12}$	$f_{13}$	$f_{11}$	$f_{12}$	$f_{13}$
Ch2	$f_{21}$	$f_{32}$	$f_{23}$	$f_{22}$	$f_{33}$	$f_{31}$	$f_{21}$	$f_{41}$	$f_{23}$
Ch1	$f_{11}$	$f_{12}$	$f_{13}$	$f_{11}$	$f_{12}$	$f_{13}$	$f_{11}$	$f_{12}$	$f_{13}$
Ch2	$f_{22}$	$f_{32}$	$f_{31}$	$f_{21}$	$f_{33}$	$f_{23}$	$f_{22}$	$f_{41}$	$f_{31}$

Fig. 2 is the RR<sup>2</sup>-tree representation of the schedule in Table I. Observe that the degree of a subtree is the block period of the fragments assigned to the subtree. It is easy to see that fragments  $\{f_{11}, f_{12}, f_{13}\}$  have block period 1, fragments  $\{f_{21}, f_{22}, f_{23}, f_{31}\}$  have block period 2, and fragments  $\{f_{32}, f_{33}, f_{41}\}$  have block period 3. The offset of  $f_{41}$ , for example, is 1, and it indicates the number of other fragments preceding it within the same block.

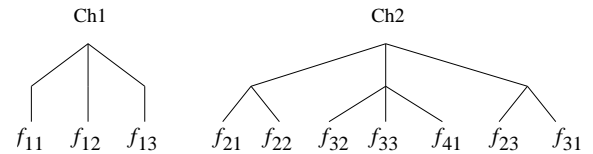


Fig. 2. RR<sup>2</sup>-tree representation of the schedule in Table I.

### III. LOWER BOUND ON STARTUP DELAY OF FSP SCHEMES

Let us first consider HW scheduling [5], [12] ( $b = m = 1$ ) with just one channel ( $c = 1$ ), hence the total available bandwidth is  $r$  (bits/sec). On this channel the entire video of duration  $D$  (sec) is broadcast repeatedly, achieving an average startup delay of  $((m - 0.5)/n)D = D/2$  for users whose arrivals follow the Poisson distribution. For the same average startup delay, any FD schedule requires bandwidth of at least  $r \ln(1 + 1/w^*) = r \ln(3) > r$  by (3). This demonstrates that at least in this special case ( $b = m = c = 1$ ), an FSP scheme can achieve a much shorter average startup delay than any FD scheme.

Recall that each page consists of  $B$  bits of data, except possibly the last page that may contain less than  $B$  bits. By Lemma 1, page  $i$  requires bandwidth  $\geq r/i$ . Therefore, an upper bound on the number of normal-sized pages that can be scheduled using bandwidth  $cr$  is given by  $h(c)$ , which is the maximum  $n$  satisfying the following inequality [5], [12]:

$$H_n = 1 + 1/2 + \dots + 1/n \leq c. \quad (5)$$

Note that when all the data of the first  $h(c)$  pages are scheduled using bandwidth at least  $H_{h(c)}$ , there may be leftover bandwidth of up to  $c - H_{h(c)}$ . In this section, we try to

<sup>2</sup>Our scheduling algorithm to be presented in Section IV is also called “Promotion”, but has nothing to do with this technique.

schedule additional fragments (of the fractional page  $P_{n+1}$ ), thus increasing duration  $D$  of a video to be broadcasted. This will result in a decreased normalized average startup delay  $T_B/2D = B/(2rD)$ . In the rest of this section we shall derive a lower bound on the normalized average startup delay that can be achieved by maximizing  $D$ .

*Lemma 2:* Given  $c$  channels, consider a sequence  $S$  of  $c$ -tuples of bits, and assume that the bits of the pages  $P_i$  ( $1 \leq i \leq n$ ) satisfy the conditions for  $S$  to be a valid FSP schedule (with delay parameter  $m$ ). As for the  $j^{\text{th}}$  bit of page  $P_{n+1}$ , it is necessary and sufficient that one of the following conditions be satisfied:

- (a) It appears at least once in every  $n + m - 1$  consecutive blocks in  $S$ , or
- (b) if it doesn't appear in  $n + m - 1$  consecutive blocks in  $S$ , then it appears within the first  $j$  bit positions of the next block.

*Proof:* First we want to prove that if the conditions of the lemma are satisfied then there is no starvation (i.e., the situation where a bit is not available at the time it is to be displayed) for any bit of  $P_{n+1}$ . Counting from the time when display starts at a block boundary, the  $j^{\text{th}}$  bit of  $P_{n+1}$  will be needed for display at bit time  $(n + m - 1)B + j$ .

In case (a) this bit appears at least once in every  $n + m - 1$  consecutive blocks, and it will have been downloaded by bit time  $(n + m - 1)B$ . In case (b) this bit may not have appeared in the first  $n + m - 1$  blocks (or in the first  $(n + m - 1)B$  bit positions) from the start of display, but at the latest it will appear in the  $n + m^{\text{th}}$  block while it is being displayed. Moreover, condition (b) guarantees that it will appear in the first  $j$  bit positions in the current block, in time for display. It is clear that if the conditions of the lemma are violated for any bit, then starvation will occur some time during the display. ■

*Theorem 3:* Let the given bandwidth be  $cr$ , where  $r$  is the display rate and  $c$  is a real number ( $\geq 1$ ). Let  $h_m(c)$  denote the maximum integer  $n$  satisfying

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

and let  $X = c - H_{n+m-1} + H_{m-1}$  ( $< 1/(n + m)$ ). The following formula gives an upper bound on the number of pages that can be scheduled in any FSP schedule (with "delay parameter"  $m$ ):

$$\frac{h_m(c) + (m - 1)X}{1 - X}. \quad (6)$$

*Proof:* In this proof we denote  $h_m(c)$  by  $n$  for simplicity. By Lemma 1 we need bandwidth at least  $r(H_{n+m-1} - H_{m-1})$  to schedule the pages in  $\{P_i \mid 1 \leq i \leq n\}$ . Therefore, the maximum bandwidth left after these  $n$  pages have been scheduled is  $rX = r(c - H_{n+m-1} + H_{m-1})$ .

Let  $p_i$  (bits) denote the amount of data in page  $P_i$ . We thus have  $p_i = B$  for  $n = 1, 2, \dots, n$ , and  $p_{n+1} < B$  for some constant  $B$ . To maximize the video duration  $D = \sum_{i=1}^{n+1} p_i/r = nB/r + p_{n+1}/r$ , we want to determine the maximum value for  $p_{n+1}$ . To maximize  $p_{n+1}$ , it is beneficial to schedule some bits of  $P_{n+1}$  in every  $n + m$  blocks, instead of at higher frequency (see Lemma 2). The first such candidate

should be the last, i.e., the  $p_{n+1}^{\text{st}}$ , bit of  $P_{n+1}$ , since condition (b) of Lemma 2 is easiest to satisfy for this bit. Clearly, the second best candidate is the second last bit of  $P_{n+1}$ , etc.

Suppose that we schedule the last  $x$  bits out of page  $P_{n+1}$  with block period  $n + m$ . By Lemma 2 (b), all these  $x$  bits must be placed within the first  $p_{n+1}$  bit positions from the beginning of a block. Since they are spread over  $n + m$  consecutive blocks, there are a total of  $(n + m)(c - (H_{n+m-1} - H_{m-1}))p_{n+1} = (n + m)Xp_{n+1}$  bit positions in  $n + m$  consecutive blocks to accommodate them. We thus should have

$$x \leq (n + m)Xp_{n+1}. \quad (7)$$

Since  $X < 1/(n + m)$ , we have  $p_{n+1} > x$ , as we should. Now the bandwidth  $rX = r(c - H_{n+m-1} + H_{m-1})$ , which is still available after the bit position  $p_{n+1}$  to the end of the block, can be used to schedule the first  $(p_{n+1} - x)$  bits of  $p_{n+1}$  in such a way that they appear in every  $n + m - 1$  blocks. Of course, we can make the period less than  $n + m - 1$ , but that would unnecessarily waste bandwidth and would not lead to the maximum  $p_{n+1}$ . Since these bits are spread over  $n + m - 1$  consecutive blocks, they are guaranteed to be downloaded in time for their display, as in the case of Lemma 2(a). Therefore, they can be placed after bit position  $p_{n+1}$  of the block in which they are placed. There are  $(n + m - 1)(B - p_{n+1})X$  bit positions to accommodate them. We thus have

$$p_{n+1} - x \leq (n + m - 1)(B - p_{n+1})X. \quad (8)$$

or

$$((n + m)X + 1 - X)p_{n+1} \leq x + (n + m - 1)BX. \quad (9)$$

Note that  $(n + m)X + 1 - X > (n + m)X$  so that (9) has smaller (positive) gradient than (7), when  $p_{n+1}$  and  $x$  are considered as vertical and horizontal axes, respectively. Note also that (7) and (9) delimit a triangular domain to the right of the  $p_{n+1}$  axis ( $x = 0$ ). Eliminating  $x$  from (7) and (8), we obtain  $p_{n+1} \leq Xp_{n+1} + (n + m - 1)XB$ , hence

$$p_{n+1} \leq \frac{(n + m - 1)XB}{1 - X}. \quad (10)$$

Eq. (6) is obtained by computing an upper bound on  $\sum_{i=1}^{n+1} p_i = nB + p_{n+1}$ , and dividing it by the amount of data in each "normal" page,  $B$ . ■

In the optimal case, we have

$$p_{n+1} = \frac{(n + m - 1)X}{1 - X}B.$$

Note that the above theorem is valid for non-integer values of  $c$  as well. The special case where  $X = 0$  was proved in [11].

*Corollary 4:* The following two formulas give lower bounds on the normalized maximum and average startup delays for the FSP schemes, respectively:

$$\frac{m(1 - X)}{h_m(c) + (m - 1)X}, \quad (11)$$

$$\frac{(m - 0.5)(1 - X)}{h_m(c) + (m - 1)X}. \quad (12)$$

*Proof:* By Theorem 3,  $(h_m(c) + (m-1)X)B/(1-X)r$  is the maximum possible video duration. Since the maximum startup delay is the time duration of  $m$  blocks, dividing  $mB/r$  (the duration of  $m$  blocks) by the maximum video duration, we obtain (11). The average startup delay is  $m - 0.5$  block time. Dividing  $(m - 0.5)B/r$  (the duration of  $m - 0.5$  blocks) by the maximum video duration, we obtain (12). ■

#### IV. APPROACHING THE LOWER BOUND ON AVERAGE STARTUP DELAY

It can be verified that (12) is minimized when  $m = 1$ . We therefore will discuss only Harmonic Block Windows (HBW) scheduling with  $m = 1$  in this section.

##### A. Main theorem

A sequence of  $c$ -tuples of fragments is said to be *block-uniform* if for all  $i$  ( $1 \leq i \leq n+1$ ) the number of fragments with block period  $i$  appearing in any block is the same. In what follows we consider only block-uniform schedules. We now state a theorem with  $m = 1$  but for a general fragment size  $b$ . Recall that Lemma 2 is valid for general  $m$ , but is relevant only to the last page. Another difference is that the following theorem is about block-uniform schedules.

*Theorem 5:* A block-uniform sequence  $S$  of  $c$ -tuples of fragments is a valid HBW schedule for  $c$  channels if and only if, for all  $i \in \{1, 2, \dots, n+1\}$ , the  $j^{\text{th}}$  fragment of  $P_i$  appears either

- (a) at least once in every  $i - 1$  consecutive blocks in  $S$ , or
- (b) once in every  $i$  consecutive blocks in  $S$ , and in the block it appears, it does so within the first  $j$  slices.

*Proof:* Let  $k$  be any non-negative integer, and consider the following subsequence of  $((i-1)b + j)$   $c$ -tuples of  $S$ :

$$F_{kb+1}, F_{kb+2}, \dots, F_{(k+i-1)b+j}. \quad (13)$$

Note that the  $c$  fragments of  $F_{kb+1}$  are transmitted in the first slice of a block, and therefore it is possible that this is the first  $c$ -tuple of fragments that a user downloads when s/he starts viewing the video. Regardless of the value of  $k$ , for any  $i$  ( $1 \leq i \leq n+1$ ) and  $j$  ( $1 \leq j \leq b$ ), the subsequence of the form (13) must contain fragment  $f_{ij}$ . Otherwise, the viewer will starve for  $f_{ij}$  when it is needed at slot time  $(i-1)b + j$  measured from the time when s/he started to view the video.<sup>3</sup> It is clear this condition is satisfied if and only if either (a) or (b) holds. ■

Ideally, we would like to schedule each fragment of page  $i$  with block period  $i$ . However, this is not always possible due to conflicts among fragments of different pages in satisfying the condition of Theorem 5 (b). We thus need to schedule some fragments of  $P_i$  with the shorter block period  $i - 1$  to satisfy the more relaxed condition of Theorem 5 (a). These fragments are said to be *promoted* from  $P_i$ .

Given a schedule  $S$ , let  $Q_i$  denote the set of all fragments scheduled with block period  $i$ , and let  $\Delta_i$  denote the set of fragments promoted from  $P_i$  to  $Q_{i-1}$ . For the schedule in

Table I, for example, we have  $Q_1 = \{f_{11}, f_{12}, f_{13}\}$ ,  $Q_2 = \{f_{21}, f_{22}, f_{23}, f_{31}\}$ ,  $Q_3 = \{f_{32}, f_{33}, f_{41}\}$ ,  $\Delta_3 = P_3 \cap Q_2 = \{f_{31}\}$  and  $\Delta_4 = P_4 \cap Q_3 = \{f_{41}\}$ . Let  $q_i$  denote the number of fragments with block period  $i$  that appears in each block of a block-uniform schedule. Thus the number of fragments in  $Q_i$  is given by  $|Q_i| = iq_i$ .

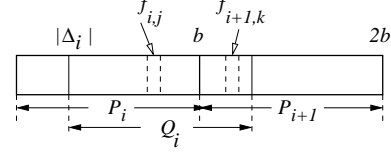


Fig. 3. The  $i^{\text{th}}$  and  $i+1^{\text{st}}$  blocks in which the fragments of  $P_i$  and  $P_{i+1}$  are to be displayed.

Fig. 3 shows two consecutive blocks, the  $i^{\text{th}}$  and  $i+1^{\text{st}}$  blocks, of schedule  $S$ . The  $i^{\text{th}}$  block consists of the fragments of  $P_i$ , i.e.,  $f_{i1}, f_{i2}, \dots, f_{ib}$  in this order, while the  $i+1^{\text{st}}$  block consists of the fragments of  $P_{i+1}$ , i.e.,  $f_{i+1,1}, f_{i+1,2}, \dots, f_{i+1,b}$  in this order. Most fragments in  $Q_i$  belong to  $P_i$ , while the remaining fragments of  $Q_i$  are those promoted from  $P_{i+1}$ . In other words,  $Q_i = (P_i - \Delta_i) \cup \Delta_{i+1}$ . Since  $q_i$  is independent of blocks in a block-uniform schedule, a fixed number of slots in each slice block is allocated to the fragments in  $Q_i$ . As there are only  $cb$  slots in each slice block, we must have

$$\sum_{i=1}^{n+1} q_i = \sum_{i=1}^{n+1} |Q_i|/i \leq bc. \quad (14)$$

In Fig. 2, it is seen that the fragments in  $Q_3 = \{f_{32}, f_{33}, f_{41}\}$  are allocated to the second subtree of the  $RR^2$ -tree for channel 2. Their block period is 3, because this subtree has degree 3. They all have offset 1, since there is one subtree to its left, and thus each of them appears in the second slot in a block. Note that fragment  $f_{41}$  satisfies condition (a) of Theorem 5, while  $f_{32}$  and  $f_{33}$  satisfy condition (b).

##### B. Promotion Algorithm for Harmonic Block Windows Scheduling

All the fragments in  $Q_i = (P_i - \Delta_i) \cup \Delta_{i+1}$  have block period  $i$ . To construct a block-uniform schedule, we distribute them equally over  $i$  consecutive blocks. However, the fragments in  $(P_i - \Delta_i)$  must satisfy Theorem 5(b), which means they must be scheduled in subtrees with no more than certain offsets. The fragments in  $\Delta_{i+1}$ , on the other hand, can be scheduled in subtrees with no offset constraint. They are spread over  $i$  successive blocks using new subtrees of degree  $i$ , but their positions within a block are not constrained in any way (see Theorem 5(a)).

We construct a schedule by means of  $c$   $RR^2$ -trees representing it. Initially, we introduce  $c$  (potential)  $c$   $RR^2$ -trees each of which has  $b$  place holders for subtrees, arranged from left to right. We fill in those place holders with subtrees as we go on. For  $j = 1, 2, \dots$ , let  $S_d[j]$  denote the  $j^{\text{th}}$  subtree of degree  $d$  that we introduce in order to schedule fragments with block period  $d$  [14]. If we put it in a place holder belonging to an  $RR^2$  tree  $T$ ,  $o_T(S_d[j])$  denotes the offset of  $S_d[j]$  in  $T$ . Consider the first subtree  $S_i[1]$  of degree  $i$ , and let it belong

<sup>3</sup>The first slot time after the display starts is numbered 1.

to one of the  $c$   $RR^2$ -trees,  $T$ . If we assign the first  $i$  fragments in  $P_i - \Delta_i$ , i.e.,  $f_{i,|\Delta_i|+1}, \dots, f_{i,|\Delta_i|+i}$ , to this subtree, then they all have block period  $i$ . We must choose the position of  $S_i[1]$  within  $T$  to satisfy  $o_T(S_d[1]) \leq |\Delta_i|$ , so that  $f_{i,|\Delta_i|+1}$  has offset at most  $|\Delta_i|$ .

For the next  $i$  fragments  $f_{i,|\Delta_i|+i+1}, \dots, f_{i,|\Delta_i|+2i}$ , we introduce the second subtree  $S_i[2]$  of degree  $i$  (say, in tree  $T'$ ) with offset  $o_{T'}(S_d[2]) \leq |\Delta_i| + i$ , etc. In general, for  $j = 1, 2, \dots$ , the “ideal”  $S_i[j]$  has offset  $|\Delta_i| + i(j-1)$ . We can make “ideal” assignment first for all the fragments in  $P_1$ , then try to do so for those in  $P_2, P_3$ , etc. We give higher priority to the fragments of earlier pages, to avoid promotion from an earlier page, which costs more bandwidth, than that from a later one. This is because the frequency (=1/period) of the promoted fragments goes up by  $i/(i-1)$ , which is larger for smaller  $i$ . Eventually, we may encounter the situation where we cannot choose such an “ideal” subtree  $S_i[j]$ , because there is no subtree left whose offset is  $|\Delta_i| + i(j-1)$ . There are two possibilities in this case: (a) If there is still a subtree position whose offset is less than  $|\Delta_i| + i(j-1)$ , then we can put  $S_i[j]$  there. (b) Otherwise, we need to increase  $|\Delta_i|$ , i.e., we need to promote more fragments from  $P_i$ , so that  $o_T(S_d[j]) = |\Delta_i| + i(j-1)$  holds, if we choose an appropriate subtree in an  $RR^2$ -tree  $T$ . To take care of the promoted fragments, we need to create a subtree of degree  $i-1$ , and promote even more fragments from  $P_i$  to fill all its leaves. If there still are fragments of  $P_i$ , we proceed to introduce  $S_i[j+1]$ , and so forth. Eventually, after introducing subtree  $S_i[n'_i]$ , all fragments of  $P_i$  are scheduled.

For a pair  $i$  ( $1 \leq i \leq n+1$ ) and  $\ell$  ( $1 \leq \ell \leq i$ ), let us define a *coset* as follows:

$$COS_{i,\ell} = \{f_{i, (|\Delta_i| + k_i + \ell)} \in P_i \mid k = 0, 1, \dots\},$$

where  $|\Delta_i|$  is the final value obtained as above. The number of subtrees introduced is given by  $|COS_{i,1}| = \lceil (|P_i| - |\Delta_i|)/i \rceil$ . If the number of fragments in  $P_i - \Delta_i$  is an integer multiple of  $i$ , then we have  $|COS_{i,1}| = |COS_{i,b}|$ . If the number of fragments in  $P_i - \Delta_i$  is not an integer multiple of  $i$ , on the other hand, then we have  $|COS_{i,1}| = |COS_{i,b}| + 1$ , in which case, some fragments from  $P_{i+1}$  are promoted to fill the leaves of the last subtree  $S_i[n'_i]$ , where  $n'_i$  was defined above. These fragments are the initial members of  $\Delta_{i+1}$ .

The above idea is the basis of our *Promotion Algorithm*, which is given in the Appendix as pseudocode. Its details are discussed in [14]. Here we explain it by means of a detailed example.

**Example** Table II shows an example, where  $c = 2$ ,  $m = 1$  and  $b = 19$ .<sup>4</sup> The two  $RR^2$ -trees, one for each channel, are represented by the two rows of the table. If subtree  $S_i[j]$  is placed in the  $(k, \ell)$  cell of the table, where  $k = 1, 2$ , it indicates that this subtree ( $S_i[j]$ ) is the  $\ell^{th}$  subtree of the  $k^{th}$   $RR^2$ -tree.

The subtrees of degree 1,  $\{S_1[j] \mid j = 1, \dots, 19\}$ , for the fragments of  $P_1$  are placed in every cell of the first row, representing channel 1. The subtrees of degree 2,  $\{S_2[j] \mid j = 1, \dots, 10\}$ , are placed in every other cell of the second

row, representing channel 2. They accommodate all the  $b = 19$  fragments of  $P_2$ . The last subtree,  $S_2[10]$ , has one empty leaf, to which we assign the promoted fragment  $f_{3,1}$ . One fragment of  $P_3$  is thus promoted, making the initial value of  $|\Delta_3| = 1$ . We now need to introduce subtree  $S_3[1]$  for fragments  $f_{3,2}, f_{3,3}, f_{3,4}$ . The ideal place for this subtree is column 2, since the first fragment,  $f_{3,2}$ , can have offset 1. There is an empty cell there, so we place  $S_3[1]$  in column 2. The next subtree is  $S_3[2]$  for fragments  $f_{3,5}, f_{3,6}, f_{3,7}$ . The “ideal” column for it is  $|\Delta_3| + 3 \times (2-1) + 1 = 5$ , but there is no empty cell in column 5. We thus shift it one column to the left into column 4. The next subtree is  $S_3[3]$  for fragments  $f_{3,8}, f_{3,9}, f_{3,10}$ . The “ideal” column for it is  $|\Delta_3| + 3 \times (3-1) + 1 = 8$ , and there is an empty cell in column 8, so we place  $S_3[3]$  there.

Proceeding as above, we place subtree  $S_3[6]$  in column 16. Now that we have created 6 subtrees of degree 3, we can accommodate 18 fragments ( $f_{3,2}$  to  $f_{3,19}$ ) in them. Now the “ideal” column for subtree  $S_4[1]$  is column 1, but the first column with an empty cell is column 6. This implies we need to promote  $|\Delta_4| = 5$  fragments  $f_{4,1}, \dots, f_{4,5}$ . This necessitates creating two additional subtrees,  $S_3[7]$  and  $S_3[8]$ , which can accommodate  $f_{4,1}, \dots, f_{4,6}$ , making  $|\Delta_4| = 6$ .  $S_4[1]$  is now placed in column 6, and it can accommodate the four fragments  $f_{4,7}, \dots, f_{4,10}$ . Since there is no empty cell left in the table, we are done. The total number of fragments scheduled is  $f = 19 \times 3 + 10 = 67$ , and its page-equivalent is  $f/b = 67/19 = 3.53 > h(3) = 3$ . In other words, we can exceed the harmonic bound. The normalized average startup delay is given by  $(b/2)/f = (19/2)/67 = 0.142$ .

Note that the distance between  $S_3[1]$  and  $S_3[2]$  in channel 2 is 2, and the distance between  $S_3[2]$  and  $S_3[3]$  is 4. None of them are exactly distance 3 apart, although the average distance is not more than 3.<sup>5</sup> This is the flexibility of Harmonic Block Windows scheduling, compared to Harmonic Windows scheduling. The total number of subtrees with 3 leaves (of the form  $S_3[j]$ ) in Table II is 8. The last two of these subtrees are used solely to schedule the six promoted fragments in  $\Delta_4$ , and therefore, they can appear anywhere in the  $RR^2$ -tree. ■

We can shift some subtrees either to the left or to the right of the “ideal” positions, as mentioned in the above example. Shifting such a subtree to the left is preferred since it doesn’t incur any additional promoted fragments, which waste bandwidth. In case there is no empty cell to the left of the ideal position, we are forced to shift it to the right with the accompanying cost of increasing  $|\Delta_i|$ .

## V. COMPARISON

The “FD Bound” and “FSP Bound” values in Table III were computed by (4) and (6) with  $m = 1$ , respectively. The ratio between these two values are shown in the fourth row, and they are almost a constant, indicating that (6) is approximately exponential. The values in the row “Promotion” are for the schedules generated by our Promotion Algorithm.

<sup>5</sup>In windows schedules, the maximum distance between any two successive appearances of page 3 must be 3.

<sup>4</sup>We have  $n = h(2) = 3$ . Page 4 is a partial page.

TABLE II

EXAMPLE OF SUBTREE ASSIGNMENT. (THE FOURTH, FIFTH AND SIXTH ROWS ARE CONTINUATIONS OF THE FIRST, SECOND AND THIRD ROWS, RESPECTIVELY.)

Index	1	2	3	4	5	6	7	8	9	10
Ch 1	$S_1[1]$	$S_1[2]$	$S_1[3]$	$S_1[4]$	$S_1[5]$	$S_1[6]$	$S_1[7]$	$S_1[8]$	$S_1[9]$	$S_1[10]$
Ch 2	$S_2[1]$	$S_3[1]$	$S_2[2]$	$S_3[2]$	$S_2[3]$	$S_4[1]$	$S_2[4]$	$S_3[3]$	$S_2[5]$	$S_3[4]$
Index	11	12	13	14	15	16	17	18	19	—
Ch 1	$S_1[11]$	$S_1[12]$	$S_1[13]$	$S_1[14]$	$S_1[15]$	$S_1[16]$	$S_1[17]$	$S_1[18]$	$S_1[19]$	—
Ch 2	$S_2[6]$	$S_3[7]$	$S_2[7]$	$S_3[5]$	$S_2[8]$	$S_3[6]$	$S_2[9]$	$S_3[8]$	$S_2[10]$	—

Note that the lower bounds are valid for all FSP schedules, regardless of whether they are block-uniform or not. The simulation results indicate that the performance of block-uniform schedules (generated by the Promotion Algorithm) approaches the lower bounds extremely closely as  $b$  is increased. In fact, they are identical up to at least three significant digits to the corresponding FSP bounds.

Fig. 4 illustrates the normalized startup delays as functions of the server bandwidth  $cr$ . Note that the lower bound on the average startup delays for the FSP schemes with  $m = 1$  lies at the bottom. The lower bound on the startup delays for any FD scheme,  $1/(e^c - 1)$  of (6), is seen as the second curve from the bottom. The figure also plots the lower bounds on the *maximum* startup delays for the FSP schemes with  $m = 1, 2$  and 4. As commented earlier, the FD schemes can be interpreted as a limiting case of the FSP schemes as  $m \rightarrow \infty$ . Therefore, the lower bound on the *maximum* startup delays for the FSP schemes approaches that for the FD schemes as  $m \rightarrow \infty$ .

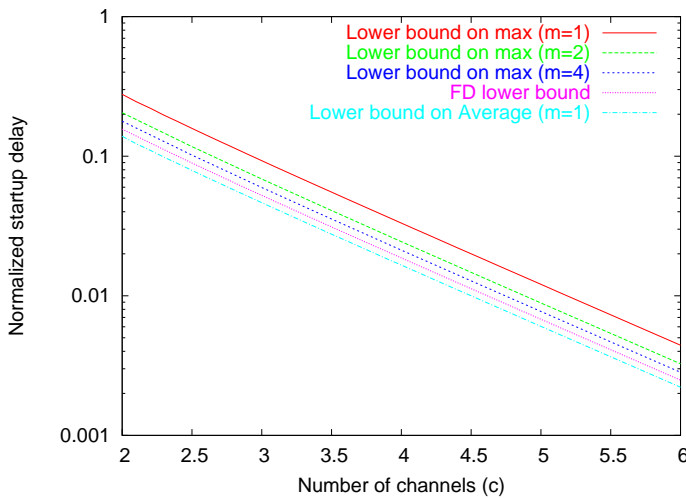


Fig. 4. Lower bounds on normalized startup delay vs. available bandwidth  $cr$ .

Fig. 5 shows the average startup delays of the schedules generated by the Promotion Algorithm for  $m = 1$  and block sizes  $b = 4, 16$  and 64, as well as the lower bound (4) on the maximum startup delay. It also plots two data points from [10] for the maximum startup delays. Observe that for all  $b \geq 16$ , the average startup delays are shorter than the general lower bound on the maximum delay for any scheme, and for  $b = 64$  the average startup delays of the schedules generated by Promotion Algorithm are practically indistinguishable from

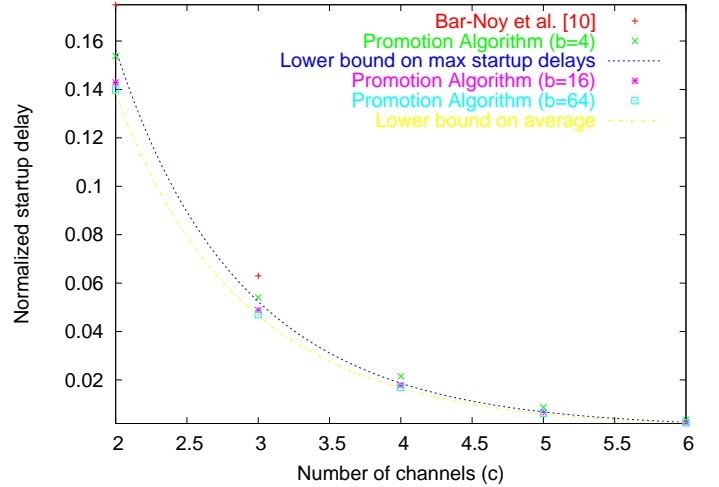


Fig. 5. Normalized startup delay vs. available bandwidth  $cr$  for schedules generated by the Promotion Algorithm ( $m = 1$ ).

the lower bound. The number of pages used in [10] for the two data points (maximum startup delays) in Fig. 5 are  $(c, \text{number of pages}) = (2, 137)$  and  $(3, 127)$ . The average startup delays of the schedules generated by our Promotion Algorithm for the case  $m = 1$  and  $b = 4$  are less than the above two data points, even though the numbers of fragments used are only 13 ( $< 137$ ) and 37 ( $< 127$ ), respectively.

## VI. CONCLUDING REMARKS

In this paper we established the tight lower bound on the average startup delay for the FSP schemes, solving one of the problems left open in [10]. We also introduced Harmonic Block Windows (HBW) scheduling, generalizing Shifted Harmonic Windows scheduling [10], [8].

We presented a family of HBW schedules which can achieve shorter average startup delay than any other currently known scheme. Although fragmenting pages may incur extra processing overhead, at least in the practical VoD application, a fragment would be still larger than an IP packet, even if each page is divided into several hundred fragments, so that it will have little impact on the overhead for packet transmission and processing.

However, achieving a shorter *average* startup delay by a FSP scheme comes at the cost of a longer *maximum* startup delay compared with FD schemes, as seen in Fig. 4.

If it is desired to start display instantaneously, we can use *prefix-caching* [15]. The initial  $T_B$  sec (or the first page) of

TABLE III  
NORMALIZED AVERAGE STARTUP DELAYS WHEN BANDWIDTH  $cr$  IS USED.

# of channels ( $c$ )	1	2	3	4	5	6	7	8
FD bound (3)	0.582	0.157	0.0524	0.0187	0.00678	0.00248	0.000913	0.000336
FSP bound (6), $m = 1$	0.5	0.139	0.0465	0.0166	0.00604	0.00221	0.000813	0.000299
Ratio	1.164	1.130	1.128	1.128	1.123	1.121	1.123	1.124
Promotion ( $b \leq 1000$ )	0.5	0.139	0.0465	0.0166	0.00604	0.00221	0.000813	0.000299

each video is cached by every user, so that if he/she wants to watch the video, it is locally available before the remaining portion of the video is downloaded using an FD or FSP scheme.

## VII. ACKNOWLEDGEMENT

We are grateful to Fei Ma of Microsoft Corporation (formerly of Simon Fraser University), for insightful discussions regarding the Promotion Algorithm in Section IV.

### Appendix: Promotion Algorithm

*Inputs:*  $c$  (number of channels),  $b$  (block size), and  $n = h(c)$

*Variables* ( $P_i$ ,  $Q_i$ ,  $S_i[j]$  and  $n_i$  are defined in the text.):

$B_i^T$  */\*\** =  $|\Delta_i|$ . Offset base of  $Q_i$  */\*\**

*avail* */\*\** # of empty cells in SST.<sup>6</sup> Maintained throughout for-loop For1. */\*\**

*credit* */\*\** # of empty leaves in last subtree created in previous round of For1 loop. Passed to next round */\*\**

*st4promo* */\*\** # of subtrees dedicated to promoted fragments from  $P_i$ . Computed in each For1 round */\*\**

*low* = 1; */\*\** leftmost non-full column. Maintained throughout for-loop For1. */\*\**

*p* */\*\** "ideal" SST column where  $S_i[j]$  should be placed */\*\**

0. *avail* =  $b \times c$ ; *credit* = 0; */\*\** Initialization */\*\**

1. For ( $i = 1$ ;  $i \leq n + 1$ ;  $i++$ ) */\*\** Start For1; process fragment set  $P_i$  */\*\**

2.  $\{ n_i = 0$ ; */\*\** Initialize  $n_i$  */\*\**

3.  $B_i^T = \text{Max}\{low - 1, credit\}$ ; */\*\** Initialize  $B_i^T$  for round  $i$  */\*\**

4. For ( $j = 1$ ;  $p = B_i^T + 1 + (j - 1)i \leq b$ ;  $j++$ ) */\*\** Start For2: Place  $\{S_i[j]\}$  in SST */\*\**

5.  $\{\text{If } low > p \text{ let } q = low, \text{ else let } q \text{ be non-full column closest to } p \text{ and not to the right of } p;$

6. Put  $S_i[j]$  in column  $q$ ;

7.  $n_i++$ ; *avail*—; Update *low*; */\*\** Reflect placement of  $S_i[j]$  in SST */\*\**

8.  $B_i^T = B_i^T + \text{Max}\{q - p, 0\}$  */\*\** Reflect offset  $o_\tau(S_i[j]) = q - 1$  in  $B_i^T$  */\*\**

9. If ( $B_i^T > credit$ ) */\*\** Start If1: Is *credit* not enough for updated  $\Delta_i$  implied by change in  $B_i^T$ ? */\*\**

10.  $\{st4promo = \lceil (B_i^T - credit) / (i - 1) \rceil$ ; */\*\** # of additional subtrees of deg  $i - 1$  needed for promoted fragments in  $\Delta_i$  */\*\**

11. If  $st4promo \geq avail$ , then exit from For1 loop;

12.  $\}$ ; */\*\** End of If1 */\*\**

13.  $\}$ ; */\*\** End of For2 */\*\**

14. *avail* = *avail* -  $st4promo$ ;  $n_i = n_i + st4promo$ ;

15. *credit* =  $n_i \times i - (b - B_i^T)$ ; */\*\** # of empty leaves in  $S_i[n_i]$ .<sup>7</sup> */\*\**

16.  $\}$ ; */\*\** end of For1 */\*\**

17. Place subtrees for promoted fragments in empty SST cells in any order;

18. Use empty cells in SST, if any, for subtrees for additional promoted fragments from  $P_i$ ; ■

## REFERENCES

- [1] A. Hu, I. Nikolaidis, and P. Beek, "On the design of efficient video-on-demand broadcast schedules," in *Proc. 7th Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 1999, pp. 262–269.
- [2] J.-F. Pärís, S. Carter, and D. Long, "Efficient broadcasting protocols for video-on-demand," in *Proc. 6th Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, July 1998, pp. 127–132.
- [3] J.-F. Pärís, "A fixed-delay broadcasting protocol for video-on-demand," in *Proc. 10th Int'l Conf. on Computer Communications and Networks*, 2001, pp. 418–423.
- [4] S. Viswanathan and T. Imielinski, "Pyramid broadcasting for video on demand service," in *Proc. IEEE Conf. on Multimedia Computing and Networking*, vol. 2417, San Jose, CA, 1995, pp. 66–77.
- [5] L. Juhn and L. Tseng, "Harmonic broadcasting for video-on-demand service," *IEEE Trans. on Broadcasting*, vol. 43, no. 3, pp. 268–271, September 1997.
- [6] K. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proc. ACM SIGCOMM*, Cannes, France, September 1997, pp. 89–100.
- [7] J.-F. Pärís, "A simple low-bandwidth broadcasting protocol for video-on-demand," in *Proc. 8th Int'l Conf. on Computer Communications and Networks (ICCCN)*, 1999, pp. 118–123.
- [8] H. Hollmann and C. Holzschere, "European patent (1991) and us patent no. 5524271 (1995)," Philips Laboratories, Brussels, Tech. Rep., 1991.
- [9] W.-L. D. Tseng and D. Kirkpatrick, "Lower bounds on average-case delay for video-on-demand broadcast protocols," in *Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, New Orleans, LO, January 2007.
- [10] A. Bar-Noy, R. Ladner, and T. Tamir, "Scheduling techniques for media-on-demand," in *Proc. 14th ACM-SIAM Symp on Discrete Algorithms (SODA)*, 2003, pp. 791–800.
- [11] L. Engebretsen and M. Sudan, "Harmonic broadcasting is bandwidth-optimal assuming constant bit rate," in *Proc. 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2002.
- [12] A. Bar-Noy and R. Ladner, "Windows scheduling problems for broadcast systems," in *Proc. 13th ACM-SIAM Symp on Discrete Algorithms (SODA)*, 2002, pp. 433–442.
- [13] T. Kameda, Y. Sun, and L. Goddyn, "A combinatorial optimization problem related to vod broadcasting," in *Proc. Int'l Symp. on Algorithms and Computation, LNCS 3827*, Hainan, China, December 2005, pp. 116–125.
- [14] F. Ma, T. Kameda, and Y. Sun, "Efficient block windows scheduling for VOD broadcasting," August 2006, unpublished manuscript.
- [15] J.-F. Pärís, S. Carter, and P. Mantey, "Zero-delay broadcasting protocols for video-on-demand," in *Proc. ACM Multimedia Conf.*, Orlando, FL, November 1999, pp. 189–197.

<sup>7</sup>Note that  $n_i \times i$  is the total number of leaves in the  $n_i$  subtrees, and  $b - B_i^T = |P_i| - |\Delta_i|$ .

<sup>6</sup>A table like Table II is called a *subtree schedule table* (SST) in [14].

### List of Figure Captions

- 1) A block consisting of  $b$  slices or  $bc$  slots.
- 2)  $RR^2$ -tree representation of the schedule in Table I.
- 3) Maximum  $p_{n+1}$  for the case  $m = 1$ .
- 4) The  $i^{th}$  and  $i + 1^{st}$  blocks in which the fragments of  $P_i$  and  $P_{i+1}$  are to be displayed.
- 5) Lower bounds on normalized startup delay vs. available bandwidth  $cr$ .
- 6) Normalized startup delay vs. available bandwidth  $cr$  for schedules generated by the Shifting algorithm ( $m = 1$ ).

### List of Table Captions

- I An example of a block-uniform schedule (rows 3 and 4 are continuations of rows 1 and 2, respectively).
- II Example of an SST. (The fourth, fifth and sixth rows are continuations of the first, second and third rows, respectively.)
- III Normalized average startup delays when bandwidth  $cr$  is used.