An Optimization Problem Related to Truncated Harmonic Broadcasting for Video-on-Demand

Tsunehiko Kameda and Yi Sun
School of Computing Science
Luis Goddyn
Department of Mathematics
Simon Fraser University, Burnaby, B.C. Canada V5A 1S6
Email: tiko@cs.sfu.ca, sunyi@cs.sfu.ca, goddyn@math.sfu.ca

***Abstract*—**

Fixed-Delay Pagoda Broadcasting **(FDPB) is a practical broadcasting scheme for video-on-demand proposed by P$\hat{a}$ris, and its prototype has been built. In order for any broadcasting scheme to be truly practical, it is crucial that the required bandwidth be minimized. Some heuristic is used to compute the optimal number s of subchannels used by FDPB to minimize the required bandwidth, but it does not always generate the optimal solution. In order to find the truly optimal s, the only known algorithm requires to try close to $0.5m$ different potential solutions for s, where m is a parameter of FDPB which can be rather large. This paper analyzes the combinatorial optimization problem involved in detail and limits the search space for the optimal s down to $\kappa\sqrt{m}$ different values, where $\kappa \approx 0.65$. This reduces the running time of a computer program to compute the optimal value for s from a few hours down to a few seconds when $m = 10,000$. This impressive reduction is more than what the ratio $0.5m/\kappa\sqrt{m}$ would suggest, because the computation time depends on the values of s tested. The technique used in this paper could be applied to other similar problems.**

Keywords: *Multimedia, Video-on-demand, broadcasting, push system, bandwidth optimization.*

***Index Terms*— Multimedia, Video-on-demand, broadcasting, push system, scheduling, bandwidth optimization**

I. Introduction

Recently, there has been much interest in the broadcast-based delivery of popular videos, in order to address the scalability issue in video-on-demand. To the best of our knowledge, Hollmann and Holzscherer first conceived the idea of segmenting a video and broadcasting the "segments" at different frequencies [3],[1] although [11] is commonly cited as the original source of the idea. In this and several other video broadcasting schemes, a video of duration D (sec) is divided into n *pages*, $P_1, P_2, \ldots, P_n$, of duration $d = D/n$ each. (Some authors use the term "segment", instead of "page", but in our terminology, a *segment* consists of several consecutive pages.) There are c *channels*, $C_1, C_2, \ldots, C_c$, of bandwidth b (bits/sec) each that are used to broadcast a video, where b is the *display rate* of the video. In general, the bandwidth of each channel may not be equal to b, but FDPB we discuss in this paper uses channels of bandwidth b. In our model, a channel consists of an infinite sequence of consecutive *slots*, each of duration d, and slots of different channels are synchronized in the sense that they start and end at the same time. A *window* of size w is a time interval of duration wd starting at a slot boundary. The transmission of a page occupies one slot in a channel.

Unaware of the almost identical earlier work of Hollermann et al.[3], Pâris et al. proposed *Pagoda Broadcasting* (PB) [9] in 1998, and later Pâris extended it to a more elaborate version called *New Pagoda Broadcasting* (NPB) [6]. In these schemes, the viewer who "tunes in" at an arbitrary time must wait until the beginning of the next slot in C_1 (which broadcasts only P_1 repeatedly), before downloading pages from any channel. When the first bit of P_1 appears in C_1, s/he starts viewing P_1, while downloading from the other channels at the same time. It is known that a necessary condition for the viewer to be able to display continuously is that P_i be broadcast at least once in every window of size $w_i = i$ [5].[2] Therefore, if n pages are assigned to c channels, then it is necessary that $H_n = 1 + 1/2 + \cdots + 1/n \leq c$ holds, where H_n is known as the n^{th} *harmonic number* [1].

The download/display policy specifies the downloading status of a segment, when the previous segment has just been completely displayed.

[1]We are thankful to Jan Korst of Philips Research Laboratory, Belgium, for bringing this little-known work to our attention.

[2]This is also a sufficient condition if each channel has bandwidth $\geq b$.

1) *Fixed-delay policy*: For each i, all of segment S_i must have been buffered in the viewer's buffer, by the time the previous segment S_{i-1} has been completely displayed.
2) *Fixed start points policy*: The viewer waits until the next slot time, before s/he starts display. Downloading starts at the same time as the display.

As commented earlier, for the schemes based on the fixed start points policy, it is necessary and sufficient to broadcast page i at least once in every window of size w_i, provided each channel has bandwidth $\geq b$. However, for the schemes based on the fixed-delay policy, this condition is necessary and sufficient regardless of the channel bandwidth.

All the schemes we have mentioned above, including PB and its variants, NPB and QHB (Quasi-Harmonic Boradcasting), adopt the fixed start points policy. Recently, Bar-Noy et al. have formulated a combinatorial problem called the *windows scheduling problem* [1]. It is a mathematical framework for the schemes based on the fixed start points policy. This problem is defined by positive integers c, and $w_1, w_2, \ldots$, where c is the number of slotted channels and a *window* of size w_i is associated with page i. A valid *schedule* assigns page i to slots such that it appears at least once in every window of w_i slots (not necessarily in the same channel). Maximizing the number of pages scheduled in a channel when the window size is fixed at $w_i = m + i - 1$ is called the *optimal truncated-Harmonic windows scheduling problem*. When $m = 1$, it is called the *optimal Harmonic windows scheduling problem*.

Fixed-Delay Pagoda Broadcasting (FDPB(m)) proposed by Pâris [7] is one of the several known schemes that adopt the fixed-delay policy.[3] The viewer initially downloads for md (sec) from all channels before s/he starts displaying the video, where m is a positive integer. In terms of the parameters we have introduced so far, FDPB(m) has the following constraints: (a) $w_i = m + i - 1$ for a given positive integer constant m, (b) for $j = 1, 2, \ldots, c$, channel C_j is divided into s_j subchannels, each subchannel consisting of every $s_j{}^{\text{th}}$ slot of the channel, (c) each page must appear in one subchannel with a fixed period, and (d) all pages allocated to each subchannel and channel must be consecutive. Our objective is to maximize the number of pages that can be scheduled in c channels, by choosing the optimal values for $\{s_j\}$, which depend on m. The author of [7] originally thought that $s = \sqrt{m}$ was a good estimate for the optimal s value, but later found $\sqrt{m} + 1$ or $\sqrt{m} + 2$ was sometimes optimal among the examples he tested. The paper [7] left open the problem of limiting the search range. Recently, Bar-Noy et al. [2] considered this problem and proposed a method whereby they could find the optimal value by testing $m/2$ different values of s.

[3] FDPB is used in a prototype VOD system reported in [10].

We analyze this dependency in detail and show that the optimal value of s_j is guaranteed to be one of roughly $0.65\sqrt{m_j}$ possible values. where m_j is the minimum required period of the first page to be assigned to C_j. We thus can avoid time-consuming exhaustive search for the optimal $\{s_j\}$.

Without loss of generality, in this paper we mainly consider just one channel and try to find the optimal value for the number s of subchannels that maximizes the total number of pages that can be packed in the channel. The result is directly applicable to optimizing s_j for each channel C_j, if there are $c > 1$ channels.

The rest of the paper is organized as follows. Section II describes a useful tool, called the *round-robin tree*, introduced in [2]. In Section III, we derive a formula for the number of pages that can be scheduled into s subchannels of a channel under the same constraints that FDPB(m) adopts. We then establish a range for s in terms of m that needs to be searched, in order to find the optimal value for s.

Finally, in Section IV, we summarize our contributions. We also propose a new method of subchanneling such that a subchannel is divided into subsubchannels, and show that this can schedule more pages into a channel.

II. Technical Preliminaries

A. Round-robin tree

In Section III, we will be making use of the *round-robin tree* [2]. Fig. 1 is an example of a 2-level round-robin tree. A round-robin tree represents a schedule as follows:

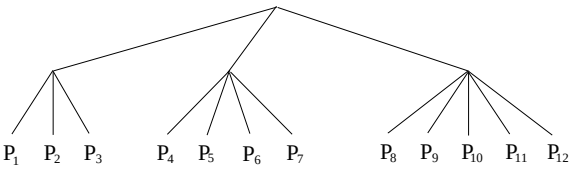

Fig. 1. A round-robin tree representation of FDPB(9) with $s = 3$ subchannels.

1) Initially the root 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 page is scheduled and the turn goes back to the root.

Applying the above rules, it is seen that Fig. 1 represents the schedule, $\langle P_1, P_4, P_8, P_2, P_5, P_9, \ldots \rangle$. Note that pages $P_1, \ldots, P_3$ have period 9, pages $P_4, \ldots, P_7$ have period 12, and pages $P_8, \ldots, P_{12}$ have period 15. It is easy to see that the period of a page is the product of the degrees of all the ancestor nodes of the leaf with which the page is associated.

Let us now examine the minimum cycle of a round-robin schedule. If the root of the underlying round-robin tree T has s children, let $S_1, \cdots, S_s$ be the round-robin schedules generated from the s subtrees, and let $c_1, c_2, \cdots, c_s$ denote their minimum cycles, respectively. Then the minimum cycle of the round-robin schedule represented by T is given by $sLCM(c_1, \cdots, c_s)$, where $LCM()$ stands for the least common multiple of the arguments.[2] Thus, the minimum cycle of the schedule represented by Fig 1 is computed by $3 \times LCM(3, 4, 5) = 180$.

B. Model

Except in Section IV, we consider only one channel of bandwidth b, equal to the display rate. The viewer initially downloads for md (sec), before s/he starts displaying the video, where d is the slot time. In order for the viewer to be able to view the video continuously, page i must be broadcast at least once in each window of size $w_i = m + i - 1$ [8]. Thus the first page ($i = 1$) is broadcast in every m^{th} slot. This uses up $1/m$ of the bandwidth, and pages 2 to m can also be scheduled in the same channel. However, there is some waste in this assignment, because page m, for example, needs to be broadcast once in every $w_m = 2m - 1$ time slots, not in every m time slots.

To make it more efficient Pâris in FDPB(m) divides each channel into s "subchannels" for some integer s. For $j = 1, 2, \ldots, s$, let subchannel j consist of every s^{th} slot.

Example 1: Let us suppose $m = 9$ and choose $s = 3$. Page 1 can be broadcast in every $w_1 = 9 + 1 - 1 = 9^{th}$ slot. Since $s = 3$, subchannel 1 consists of every 3rd slot, and page 1 needs only 1/3 of it. Thus pages 2 and 3 can also be broadcast in subchannel 1. These three pages will each have period 9, i.e., each of them will appear in every 9 time slots. Page 4 ($i = 4$) needs to be broadcast at least once in every $w_4 = m + 4 - 1 = 12$ time slots. Thus, pages 4, 5, 6, and 7 fit in subchannel 2, and these four pages will each have period 12. Similarly, page 8 ($i = 8$) needs to be broadcast in every $w_8 = m + 8 - 1 = 16$ time slots, and hence pages 8 to 12 fit in subchannel 3. Thus these five pages will each have period 15. In summary, by dividing a channel into three subchannels, we can now pack 12, instead of 9, pages in a channel. Fig. 1 shown earlier is a round-robin tree representation for FDPB(9) with 3 subchannels. ■

In order for page 1 to appear within every window of size $w_1 = m$, it must appear in at least every $\lfloor m/s \rfloor$ slots of a subchannel. This also implies that the first $n_1 = \lfloor m/s \rfloor$ pages can be scheduled in the first subchannel in such way that they all have the same period that is no greater than m. For $k = 1, 2, \ldots, s$, let n_k denote the maximum number of pages that can be scheduled in subchannel k. As we just saw above, $n_1 = \lfloor m/s \rfloor$. Now that the first n_1 pages have been scheduled in subchannel 1, the next page, i.e., the $n_1 + 1^{st}$ page must have period at most $m + (n_1 + 1) - 1$. We thus have $n_2 = \lfloor (m + n_1)/s \rfloor$ We can generalize this to get the following formula for n_k:

$$n_k = \lfloor (m + n_1 + n_2 + \cdots + n_{k-1})/s \rfloor \qquad (1)$$

Let $n(m, s)$ denote the total number of pages assigned to the s subchannels, i.e.,

$$n(m, s) = \sum_{k=1}^{s} n_k. \qquad (2)$$

III. Optimization for Truncated Harmonic Scheduling

A. Problem

Fig. 2 plots $n(100, s)$ by varying s in the range $1 \leq s \leq 100$ (the rugged curve). It is seen that $s = 10$ maximizes $n(100, s)$. In Fig. 3, $n(m, s)$ is plotted for many different values of m, where $m = 1, 2, \ldots, 130$. For each value of m, a curve is drawn by varying s within the range $1 \leq s \leq m$. This can be considered as exhaustive search by which to find the optimal s that maximizes $n(m, s)$. The light-gray curves in Fig. 3 correspond to $m = 9, 21, 51$ and 128. (The reason why these four curves are highlighted will be explained later in Section IV.) Note that the optimal value of s that maximizes $n(m, s)$ grows with m. Our main interest in this paper is to analyze the dependency of the optimal value of s on m in the hope of finding the optimal value without resorting to exhaustive search.

As we observed in Fig. 1, the schedule that FDPB(m) generates can be represented by a 2-level round-robin tree [2]. The root of this tree has degree s and its s subtrees have degrees, n_1, $n_2, \ldots$, n_s, respectively. We first prove the correctness of FDPB(m) in our current framework. In reference to Eq. (1), for $k = 1, \cdots, s$, we

Fig. 2. The rugged curve shows $n(100., s)$, when the number of subchannels s is varied from 1 to $m = 100$. The smooth curve is an approximation using Eq. (4) with $\bar{r}/s = 0.5$.

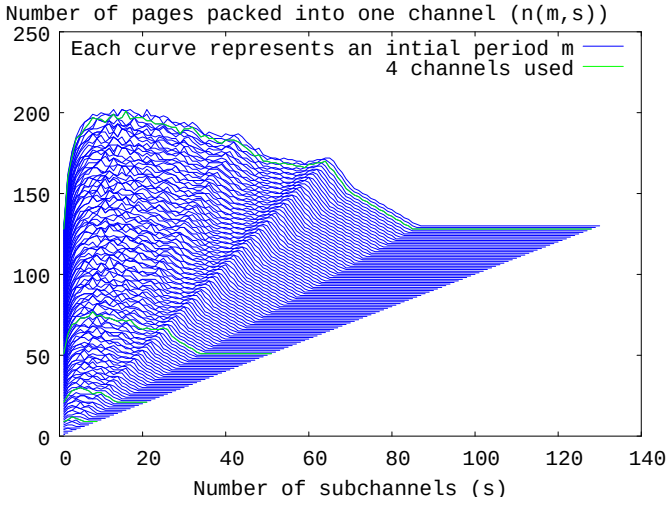

Fig. 3. The numbers of pages scheduled in one channel. Each curve corresponds to a different period for the first page, i.e., $m = 1, 2, \ldots, 130$. The number of subchannels s is varied from 1 to m.

define r_k $(0 \le r_k \le s-1)$ by

$$m + \sum_{j=1}^{k-1} n_j = sn_k + r_k. \tag{3}$$

Lemma 1: Suppose n_k is computed by Eq. (1) for $k = 1, 2, \ldots, s$ and the next n_k consecutive pages are broadcast on subchannel k, starting with page 1 on subchannel $k = 1$. Then for any i $(1 \le i \le n(m, s))$, page i has period at most $m + i - 1$.

Proof: Let T be a 2-level round robin tree and let $T_1, T_2, \ldots, T_s$ be the subtrees corresponding to the s subchannels. For a given k, let $i = \sum_{j=1}^{k-1} n_j + 1$. Then page i is the leftmost leaf of subtree T_k. Recall from Section II that all leaves of T_k have period sn_k. By Eq. (3), we have $m + i - 1 \ge sn_k$. Thus the lemma is true for the leftmost leaf of subtree T_k for any k. Since all other leaves of subtree T_k represent higher numbered pages with the same period, the lemma is also true for them. ■

In order to find the optimal value of s that maximizes $n(m, s)$ by differentiation, we try to approximate $n(m, s)$ by a function that doesn't contain any floor function. Let $\bar{r}$ denote the average of $\{r_k \mid k = 1, 2, \ldots, s\}$ in Eq. (3).

Lemma 2: The total number of pages that can be assigned to the s subchannels of a channel is approximated by the following formula when s $(< m)$ is large:

$$n(m, s) \approx (m - \bar{r}) \left((1 + \frac{1}{s})^s - 1 \right). \tag{4}$$

Proof: Let $n_0 = m$ and rewrite Eq. (3) as follows:

$$r_k = \sum_{j=0}^{k-1} n_j - sn_k,$$

and hence

$$\frac{r_k}{s}(1+1/s)^{s-k} = (1+1/s)^{s-k} \sum_{j=0}^{k-1} \frac{n_j}{s} - (1+1/s)^{s-k} n_k.$$

Summing this from $k = 1$ to s, we get

$$\begin{aligned} &\sum_{k=1}^{s} \frac{r_k}{s}(1 + 1/s)^{s-k} \\ &= \sum_{k=1}^{s} (1 + 1/s)^{s-k} \sum_{j=0}^{k-1} \frac{n_j}{s} - \sum_{k=1}^{s} (1 + 1/s)^{s-k} n_k. \end{aligned}$$

We can rewrite the right hand side as follows:

$$\begin{aligned} &\sum_{j=0}^{s-1} \frac{n_j}{s} \sum_{k=j+1}^{s} (1 + 1/s)^{s-k} - \sum_{k=1}^{s} (1 + 1/s)^{s-k} n_k \\ &= \sum_{j=0}^{s-1} n_j [(1 + 1/s)^{s-j} - 1] - \sum_{k=1}^{s} (1 + 1/s)^{s-k} n_k \\ &= m[(1 + 1/s)^s - 1] - n(m, s). \end{aligned}$$

We thus obtain

$$n(m, s) = m[(1+1/s)^s - 1] - \sum_{k=1}^{s} \frac{r_k}{s}(1+1/s)^{s-k}. \tag{5}$$

We now estimate $n(m, s)$ for large s. For sufficiently large s, we assume that the remainders $r_1, \cdots, r_s$ are

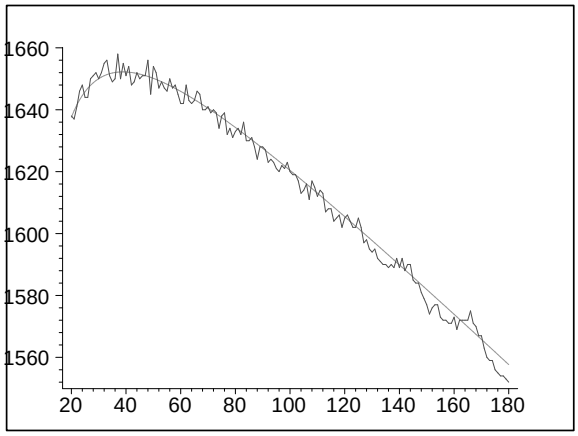

Fig. 4. The horizontal axis represents the number of subchannels (s) and the vertical axis represents $n(1000,\ s)$.

uniformly distributed in the range $[0,\ s-1]$. Substituting $r_i = \bar{r}$ into Eq. (5), we obtain Eq. (4). ■

Fig. 4 shows the maximum numbers of pages that can be scheduled in one channel when $m = 1000$ and the number of subchannels s is varied from 20 to 180. The smooth curve is an approximation using Eq. (4) with $\bar{r} = (s-1)/2$, i.e., half the possible range.

B. Solution

Lemma 3: When s is large, the normalized waiting time is lower bounded approximately by

$$\frac{1}{1-\bar{r}/m} \times \frac{1}{e-1} \tag{6}$$

for some $\bar{r}$ in the range $0 \le \bar{r} < s-1$.

Proof: Since the initial waiting time is md, by Lemma 2 the normalized waiting time is given by

$$\begin{aligned} \frac{md}{n(m,s)d} &\simeq \frac{m}{(1+1/s)^s\,(m-\bar{r}) - (m-\bar{r})} \\ &\ge \frac{1}{1-\bar{r}/m} \times \frac{1}{e-1}. \end{aligned}$$

■

Corollary 4: For large s, the total number of pages $n(m,s)$ can be bounded as follows:

$$(m-s+1)\left((1+\frac{1}{s})^s - 1\right) \le n(m,s) \le m\left((1+\frac{1}{s})^s - 1\right).$$

Proof: Follows directly from Eq. (4) by setting $\bar{r} = s-1$ (for the lower bound) and $\bar{r} = 0$ (for the upper bound). ■

In order to find the optimal value of s that maximizes $n(m,s)$, one need to evaluate Eq. (2) for s ranging $2 \le s \le m/2$ [2]. This is time-consuming when m gets large. In what follows, we show that the optimal solution $s_{\rm opt}$ can be found within a region containing $O(\sqrt{m})$ possible values of s. The following theorem shows how $s_{\rm opt}$ grows with m for large m. The following theorem was conjectured in [7] and experimentally verified in [4].

Theorem 5: The optimal number of subchannels $s_{\rm opt}$ grows linearly with $\sqrt{m}$.

Proof: Let $\bar{r} = a(s-1)$ in Eq. (4), where the range of parameter a is $0 < a < 1$. The optimal number of subchannels s satisfies the following differential equation:

$$\frac{\partial n(m,s)}{\partial s} = 0.$$

Therefore, we have

$$\begin{aligned} (1+1/s)^s \left[ln(1+1/s) + \tfrac{s^2(1/s-(s+1)/s^2)}{s+1}\right](m-a(s-1)) \\ -a\left((1+1/s)^s - 1\right) = 0. \end{aligned} \tag{7}$$

If s is sufficiently large, we can simplify the above equation as follows:

$$\frac{em/2 + \frac{23}{24}ea}{s^2} - (e-1)a \approx 0$$

By solving the above quadratic equation, using the assumption $s^2 \gg a$, we obtain

$$s_{\rm opt} \approx \sqrt{\frac{em}{2a(e-1)}}. \tag{8}$$

■

$n(m,s)$ is not a smooth function in s, as we saw in Fig. 4. In order to limit our search space for the optimal s, we need to bound the parameter $a = a(m,s)$ which is a function of m and s. We want to determine

$$a^{up} = Sup\{a(m,s)\}$$

and

$$a^{low} = Inf\{a(m,s)\}.$$

Since $a < 1$, we obviously have $a^{up} < 1$. The problem is how to bound a^{low}. Fig. 5 gives the computation results for $a(1000,\ s)$ for s in the range $1 \le s \le 1000$.

In order to find a^{low} for given m, we want to investigate how the remainder changes in the optimal region, i.e., in the vicinity of $s = \sqrt{m}$. Fig. 6 illustrates the case where $m = 10,000$ and the number of subchannels $s = 100$. The x-axis represents the 100 subchannels, i.e., subchannels $k = 1$ to $k = 100$. Each number in Fig. 6 is n_k of Eq. (1) for some k, and its height represents r_k of Eq. (3). One striking feature in Fig. 6 is the presence of quadratic curves, half a parabola near the origin and another complete parabola in the middle of the graph. Let us examine the cause of this phenomenon.

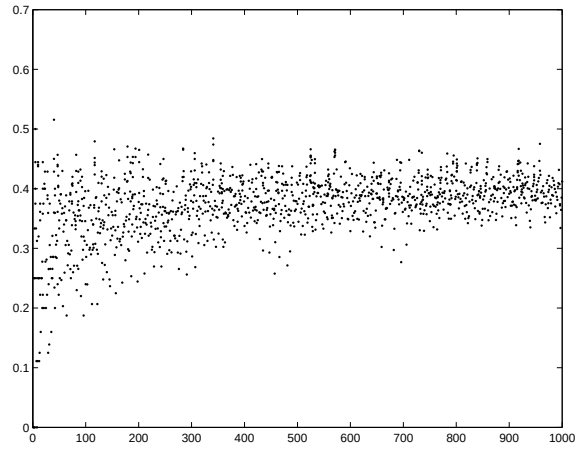

Fig. 5. a vs. the number of subchannels s ($m = 1000$).

At the bottom of the left parabola, $n_k = 100$, which equals the number of channels s being used, appears, where k is some number that is not important for our discussion. From the formula for n_k in Eq. (1), this means that

$$n_k = \left\lfloor \frac{X}{s} \right\rfloor = s.$$

Let the corresponding remainder $r_k = R$ ($R = 0$ in Fig. 6.) Then X in the above equation can be expressed as $X = s^2 + R$.[4] Let us now compute n_{k+1}, n_{k+2}, etc.

$$n_{k+1} = \lfloor (s^2 + s + R)/s \rfloor = s + 1, \quad r_{k+1} = R.$$

$$\begin{aligned} n_{k+2} &= \lfloor (s^2 + s + (s+1) + R)/s \rfloor = s + 2, \\ r_{k+2} &= R + 1. \end{aligned}$$

$$\begin{aligned} n_{k+3} &= \lfloor (s^2 + s + (s+1) + (s+2) + R)/s \rfloor \\ &= s + 3, \\ r_{k+3} &= R + 1 + 2. \end{aligned}$$

$$\begin{aligned} n_{k+4} &= \lfloor (s^2 + s + (s+1) + (s+2) + (s+3) + R)/s \rfloor \\ &= s + 4, \\ r_{k+4} &= R + 1 + 2 + 3. \end{aligned}$$

We now see the trend and can compute

$$r_{k+j} = R + \frac{j(j-1)}{2}. \tag{9}$$

If we started with $n_k > s$, then r_{k+j} would be a larger function of j. Obviously these remainders are the smallest if $R = 0$. This suggests that the "worst case" which leads to the smallest estimate for a^{low} occurs around $n_k = s$, where remainders grow slowly as we saw above. Of course, it is possible that $r_h = 0$ for some h elsewhere, but r_{h+1}, r_{h+2}, etc., quickly grow.

[4]For the parabola in the middle of Fig. 6, $X = 2s^2 + R'$.

The dotted line in Fig. 6 represents the difference $r_{k+1} - r_k (\bmod s)$ for $k = 1, 2, \ldots, s$ (=100). The lower horizontal line is the average of all the remainders $\{r_k \mid k = 1, 2, \ldots, s\}$, while the upper horizontal line is at height $(s-1)/2$, which would be the average of the remainders if they were randomly distributed between 0 and $s - 1$.

Lemma 6: In the optimal region the average remainder $a^{low} > \frac{1}{3}$.

Proof: Fix the value s near $\sqrt{m}$, and assume the worst case, i.e., r_k has the form $r_{k+j} = cj^2$ for some constant c. Thus each remainder r_k takes one of the $n+1$ values,

$$0, c, 2^2c, \ldots, n^2c,$$

where n is the largest integer satisfying $n^2c \le s - 1$, because any remainder $r_k \le s - 1$. In other words, for $k = 1, 2, \ldots, s$, $1/(n+1)$ of those remainders have value 0, $1/(n+1)$ of those remainders have value c, $1/(n+1)$ of those remainders have value 2^2c, etc. Then the average value of all the s remainders is given by

$$\frac{1}{n+1} \sum_{j=0}^{n} cj^2 = cn(2n+1)/6 \approx \frac{s-1}{3},$$

assuming n is large. This implies that $a^{low} \ge 1/3$. ■

We thus have the following theorem:

Theorem 7: [4] If m is large, the optimal number of subchannels $s_{\rm opt}$ that maximizes $n(m, s)$ can be bounded as follows:

$$\sqrt{\frac{em}{2(e-1)}} \le s_{\rm opt} \le \sqrt{\frac{3em}{2(e-1)}}.$$

Proof: The lower bound is obtained by plugging the upper bound of $a^{up} = 1$ into and the upper bound is obtained by plugging the lower bound of $a^{low} = 1/3$ into Eq. (8). ■

The above theorem implies that the search space can limited to a rather small range, allowing us to avoid a near-exhaustive search. Note that $e/2(e-1) = 0.791$, $\sqrt{e/2(e-1)} = 0.889$, $3e/(e-1) = 2.37$, and $\sqrt{3e/(e-1)} = 1.54$. Therefore, only $\sqrt{3e/2(e-1)}m - \sqrt{e/2(e-1)}m = 0.65m$ different values of s need be tested to find $s_{\rm opt}$.

Fig. 7 below plots the optimal number of subchannels, $s_{\rm opt}$, computed by exhaustive search, varying s from 1 to m to find $s_{\rm opt}$, for each period m of the first page in the range up to 10,000. It is observed that the majority of the date points are within are area bounded by $\sqrt{m} - 3$

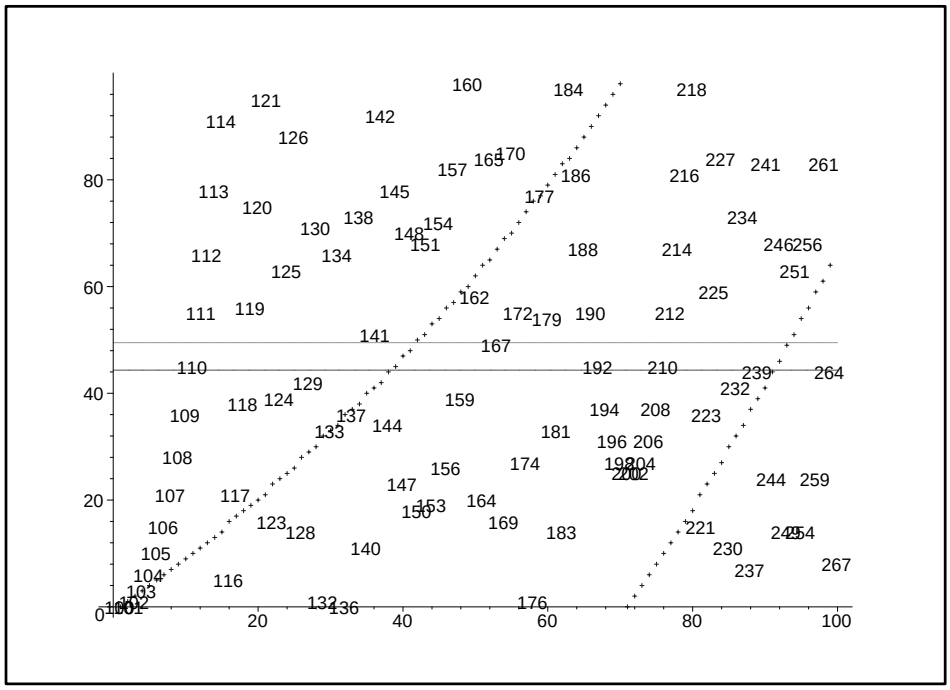

Fig. 6. Distribution of remainders $\{r_k\}$.

and $\sqrt{2.37m} + 6$. Our lower bound of $\sqrt{0.791m}$ given by Theorem 7 could be tightened a little. Due to the assumption $s \gg 1$ made to derive Eq. (4), our bounds are not valid for small values of s.

By reducing the range of search from $s \in [1, m/2]$ to $s \in [\max\{\sqrt{m} - 3, 1\}, \sqrt{2.37m} + 6]$, the execution time of our search program for all values of m between 1 and 10,000 went down rather drastically from several hours to a few seconds.[5] This range is roughly $(\sqrt{2.37} - 1)\sqrt{m} \approx 0.54\sqrt{m}$.

The middle curve in Fig. 7 represents $s = \sqrt{1.58m}$, and it appears that this curve lies roughly in the middle of the data points. We now plug $s = \sqrt{1.58m}$ into Eq. (4), and plot it in Fig. 8 below, which shows a rather good fit (within 1% error). The bottom curve shows the actual data, and the top curve shows the analytical estimate based on the assumption that $s = \sqrt{1.58m}$ is optimal. The two curves are hardly distinguishable.

[5]We used a Java program running on a Pentium II CPU.

IV. Discussions

So far we concentrated just on one channel and tried to pack as many pages in it as possible, by optimizing the number of subchannels. Let us now consider the general case, where we have c channels, $C_1, C_2, \ldots, C_c$. Suppose the maximum period of the initial page for channel C_1 is m_1. Then we have $m_2 = m + (i_1 + 1) - 1$ for first page of channel C_2, if pages 1 to i_1 are packed into channel C_1. Thus by varying the parameter m in our previous analysis, we can determine how many pages can be packed into the second, third, ... channels.

As commented earlier, the light-gray curves in Fig. 3 correspond to $m = 9, 21, 51$ and 128. The curve for $m_1 = 9$ reaches its peak 12 at $s = 3$, which implies that 12 pages can be packed into C_1 if the period of the first page is 9 and three subchannels are used. Thus page 13 is the first page to be packed in C_2, and therefore we should look at the curve for $m_2 = 9 + 13 - 1 = 21$. This curve has the peak value of 30, and thus 30 pages can be packed into C_3, and so forth.

Hollermann and Holzscherer conceived a scheme similar to FDPB(m), which we call *Hollermann-Holzscherer*

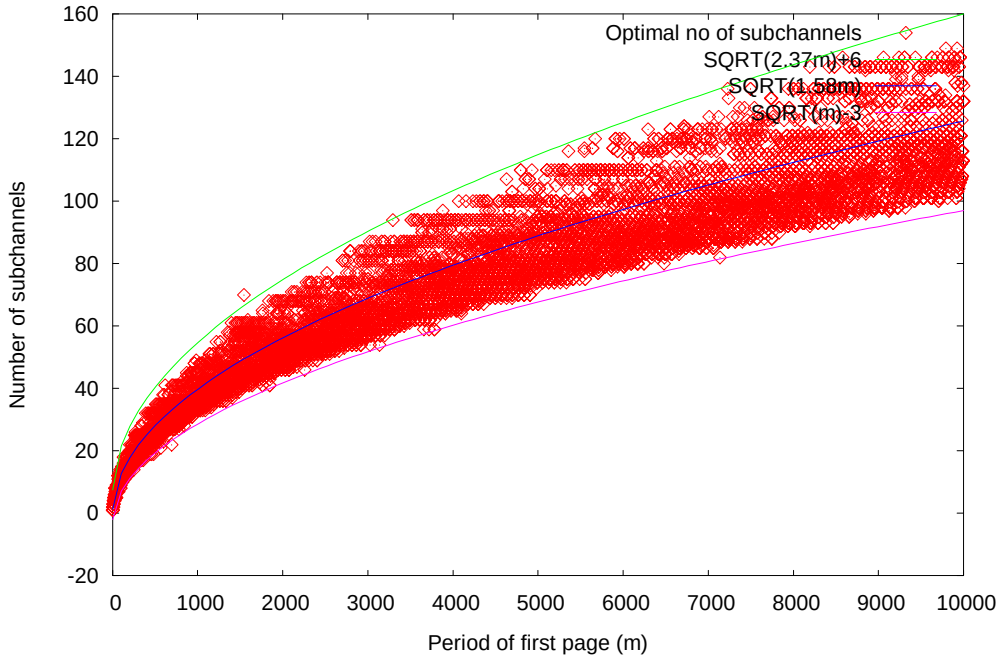

Fig. 7. The optimal number of subchannels (s_{opt}) vs. the period (m) of the first page. The actual optima are plotted as data points.

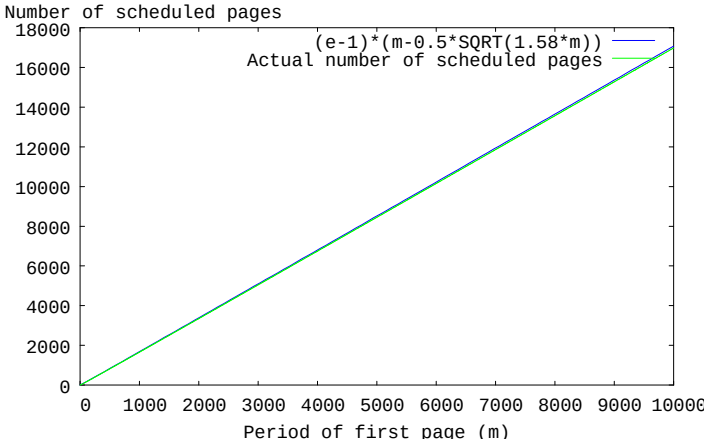

Fig. 8. Number of pages scheduled in one channel vs. the period of the first page.

Broadcasting, HHB(m) [3]. The main difference between them is that FDPB(m) is based on the fixed-delay policy, while HHB(m) is based on the fixed start points policy. Any schedule for FDPB(m) is a schedule for HHB(m) and vice versa. The waiting time for FDPB(m) is always md, while the *average* waiting time for HHB(m) is $(m - 0.5)d$, assuming Poisson arrivals. Therefore, if the average waiting time is used as the performance metric, HHB(m) slightly outperforms FDPB(m). Another difference is that the slots in different channels need not be synchronized in FDPB(m), while the synchronization is essential for HHB(m).

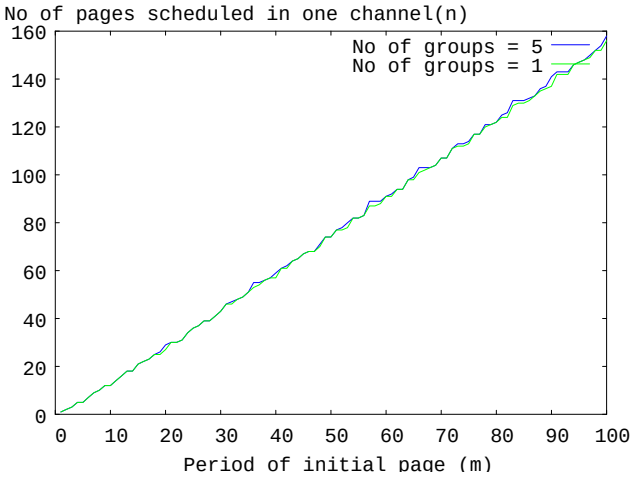

Fig. 9. Maximum number of pages scheduled in one channel vs. the period of the first page: The top curve shows the case where up to five groups were considered and best number was chosen, and the bottom curve shows the case where no grouping was used.

Recursive subchanneling: We now try recursive subchanneling. Namely, we first divide a channel into g subchannels (or groups) of equal bandwidth as before,

and then further divide each of the g subchannels into a different number of subsubchannels. As Fig. 9 shows, we are able to fit slightly more pages into a channel for some values of m. Recursive subchanneling becomes more beneficial for larger values of m.

ACKNOWLEDGEMENT

We thank the members of the Distributed Computing Laboratory in the School of Computing Science at Simon Fraser University for stimulating discussions.

REFERENCES

[1] A. Bar-Noy and R.E. Ladner, Windows scheduling problems for broadcast systems, *Proc. 13^{th} ACM-SIAM Symposium on Discrete Algorithms*, SODA 2002, pp. 433–442.

[2] A. Bar-Noy, R.E. Ladner, and T. Tamir, Scheduling Techniques for Media-on-Demand, *Proc. 14^{th} ACM-SIAM Symposium on Discrete Algorithms*, SODA 2003, pp. 791–800.

[3] H.D.L. Hollmann and C.D. Holzscherer, *Philips Tech. Rept.* 1991, European Patent (1991) and US patent No. 5524271 (1995).

[4] T. Kameda and Y. Sun, Optimal truncated-Harmonic windows scheduling for broadcast systems, Technical Report CMPT-TR 2003-10, School of Computing Science, SFU, September 2003.

[5] L. Juhn and L. Tseng, Harmonic broadcasting protocols for video-on-demand service, *IEEE Trans. on Broadcasting 43*, Sept. 1997, pp. 268–271.

[6] J.-F. Pâris, A simple low-bandwidth broadcasting protocol for video-on-demand, *Proceedings of the 8^{th} International Conference on Computer Communications and Networks* (*ICCCN'99*), Boston-Natick, MA, Oct. 1999 pp. 118–123.

[7] J.-F. Pâris, A fixed-delay broadcasting protocol for video-on-demand, *Proc. 10^{th} Int'l Conf. on Computer Communications and Networks*, 2001, pp. 418–423.

[8] J.-F. Pâris, S.W. Carter, and D.D.E. Long, Efficient Broadcasting Protocols for Video on Demand, *6th Intern'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'98)*, pp. 127–132, July 1998.

[9] J.-F. Pâris, S.W. Carter, and D.D.E. Long, A low-bandwidth broadcasting protocol for video-on-demand, *Proc. 7^{th} Int'l Conf. on Computer Communications and Networks (ICCCN '98)*, Oct. 1998, pp. 690–697.

[10] 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.

[11] 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.