

Optimal Truncated-Harmonic Windows Scheduling for Broadcast Systems

Tsunehiko Kameda^{*}
School of Computing Science
Simon Fraser University
Burnaby, B.C. Canada V5A 1S6
tiko@cs.sfu.ca

Yi Sun
School of Computing Science
Simon Fraser University
Burnaby, B.C. Canada V5A 1S6
sunyi@cs.sfu.ca

ABSTRACT

The optimal windows scheduling problem is defined by the number of slotted channels (c) and the window sizes (w_1, w_2, \dots). The transmission of a page occupies one slot in a channel, and a schedule assigns page i to slots such that it appears in every window of w_i slots. We address a special case of this problem with the following constraints: (a) $w_i = m + i - 1$ for a given positive integer constant m , (b) for $j = 1, 2, \dots, c$, channel C_j is divided into s_j subchannels, each subchannel consisting of every s_j^{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 each channel must be consecutive. We want to maximize the number of pages that can be scheduled in c channels, by choosing the optimal values for $\{s_j \mid j = 1, \dots, c\}$. We show that the optimal value of s_j is guaranteed to be one of roughly $0.6\sqrt{p}$ possible values, where p is the window size of the first page to be assigned to C_j . We thus can avoid time-consuming exhaustive search for the optimal $\{s_j\}$. Using the above results, we compare the bandwidth requirements of *Finite-Delay Pagoda Broadcasting* and *Hollermann-Holzschereer Broadcasting* schemes with those of some other VOD broadcasting schemes for a given average waiting time. It is shown that concurrent downloading can in general lead to less bandwidth requirement than segment preloading for a given average waiting time, and in fact *Quasi-Harmonic Broadcasting* requires the least bandwidth among all the known schemes.

Categories and Subject Descriptors

H3.5 [Online Information Services]: Miscellaneous

^{*}This work was supported by the Natural Sciences and Engineering Research Council of Canada.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

General Terms

Theory, performance

1. INTRODUCTION AND PRELIMINARIES

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 Holzschereer first conceived the idea of segmenting a video and broadcasting the “segments” at different frequencies [2],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, \dots, 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. See Sec. 3.) c channels, C_1, C_2, \dots, C_c , of bandwidth b (bits/sec) each are used to broadcast a video, where b is the *display rate* of the video. In our model, a channel consists of an infinite sequence of consecutive time slots, each of duration d , and slots of different channels are synchronized in the sense that they start and end at the same time.

Unaware of the almost identical earlier work of Hollermann et al.[2], Pâris et al. proposed *Pagoda Broadcasting* (PB) [8] in 1998, and later Pâris extended it to a more elaborate version called *New Pagoda Broadcasting* (NPB) [5]. 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 , he starts viewing p_1 , while downloading from the other channels at the same time. In practice, it is likely that an entire frame needs to be buffered in the *set-top box* (STB) before it can be displayed, which would cause a delay of only one frame time (about $1/30$ to $1/24$ of a second). We ignore this detail in this paper. It is known that the necessary and sufficient condition for the viewer to be able to display continuously is that p_i be broadcast in every window of size $w_i = i$ (not necessarily in the same channel) [4]. Therefore, if n pages are assigned to c channels, then it is necessary that $H_n = 1 + 1/2 + \dots + 1/n \leq c$ holds, where H_n is known as the n^{th} *harmonic number* [1].

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

Clearly, the maximum and minimum waiting times for the viewer are 0 and d , respectively, and the average waiting time is given by $d/2$. Since the ratio of the average and maximum waiting times over the duration of the video are $(d/2)/D = 1/2n$ and $d/D = 1/n$, respectively, it is desired to maximize n . Note that the maximum n is independent of the video duration D , and once the maximum n is determined, then the time slot size d can be computed by $d = D/n$.

Recently, Bar-Noy et al. have formulated optimal Pagoda Broadcasting as a combinatorial optimization problem [1], called the *optimal harmonic windows scheduling problem*, which is defined by positive integer parameters c and $w_i = 1/i$. In what follows, we sometimes refer to w_i as the *window size*. Table 1 below shows the numbers of pages that can be scheduled in c channels [1]. The upper bounds listed under U. Bound1 in the table are obtained from the maximum n such that $H_n \leq c$, while those listed under U. Bound2 are improved upper bounds derived by Tseng et al. [10]. The RFS values are achieved by the heuristic algorithm called *Recursive Frequency Splitting* in the same paper [10], which was independently discovered by Bar-Noy et al. and named *Greedy* [1]. The last row shows the n values attained by NPB [5].

# of channels (c)	1	2	3	4	5	6	7	8
U. Bound 1	1	3	10	30	82	226	615	1673
U. Bound 2	1	3	10	30	80	220	604	1650
Best known	1	3	9	28	77	211	570	1573
RFS	1	3	9	25	73	201	565	1522
NPB	1	3	9	26	66	172	442	499

Table 1: The number of pages schedulable on c channels.

In *Fixed-Delay Pagoda Broadcasting* (FDPB(m)) proposed by P aris [6], the viewer initially downloads for md (sec) from all channels before he starts displaying the video, where m is a positive integer.² In a similar scheme, which we call *Hollermann-Holzscherer Broadcasting*, HHB(m), the viewer starts downloading when the next slot starts after the tune-in time [2]. At the end of $m - 1$ slots from there, the viewer starts to display the video. At this time, page p_1 is either about to start in C_1 or has already been buffered. To see the difference between FDPB(m) and HHB(m), note that in the case where $m = 1$ HHB(1) is the same as PB, while FDPB(1) is not. FDPB(m) preloads³ each segment before displaying it. In PB and HHB(1), the waiting time is not always a fixed amount d , but is a uniform random variable with range $[0, d]$ whose average is $d/2$.

In both FDPB(m) and HHB(m), p_1 is broadcast in every window of size $w_1 = m$ in a channel. In general, p_i needs to be broadcast in every window of size $w_i = m + i - 1$. 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 scheduling problem*. To solve this problem, FDPB(m) divides channel C_j into s_j subchannels, for some integer s_j , such that each subchannel is allocated every s_j^{th} slot in that channel. Due to implementation considerations, FDPB(m) imposes the constraint that all the pages scheduled in each subchannel and each channel must be consecutive. Clearly, any schedule for FDPB(m) is a

²FDPB is used in a prototype VOD system reported in [9].

³Some authors use this term to mean *prefix caching*, i.e., downloading an initial segment prior to the display of any part of the video.

schedule for HHB(m) and vice versa. The main differences are that (a) the slots in different channels need not be synchronized for FDPB(m), while the synchronization is essential for HHB(m), and (b) the waiting time for FDPB(m) is always md , while the *average* waiting time for HHB(m) is $(m - 0.5)d$. In this paper we use the average waiting time as a performance metric, so that HHB(m) slightly outperforms FDPB(m).

In Section 2, we first 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 $p = m + i - 1$ that needs to be searched. 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.

In Section 3, we define two downloading policies used by the existing broadcasting schemes: *fixed-delay* and *fixed start points* policies. We also introduce a useful graphical tool called *download-display diagram* (DDD for short).

In Section 4, we derive the bandwidth requirements of HHB(m) and FDPB(m) based on the analysis in Section 2, compare them with those of PB and *Quasi-Harmonic Broadcasting* (QHB(q)) [7],⁴ and see how they fare relative to the lower bounds for the fixed-delay and fixed start points policies. It will be shown that QHB(q) has the least bandwidth required to achieve a given *average waiting time*, and that it requires less bandwidth than any scheme that adopts the fixed-delay policy. Finally, in Section 5, we summarize our contributions.

2. OPTIMIZATION FOR TRUNCATED HARMONIC SCHEDULING

Suppose the maximum period of the initial page is p , and let s denote the number of subchannels (in a channel) used. In this section, we discuss a generic channel without distinguishing among different channels, since the solution to the generic case is applicable to all channels by varying p . We have $p = m + i - 1$, if the initial page is page i of the video, as we mentioned above. Since the channel is divided into s subchannels, page i must appear in at least every $\lfloor p/s \rfloor$ slots of a subchannel. For $k = 1, 2, \dots, s$, let n_k denote the maximum number of pages that can be scheduled in subchannel k . We can derive a formula for n_k as follows:

$$\begin{aligned}
 n_1 &= \lfloor p/s \rfloor \\
 n_2 &= \lfloor (p + n_1)/s \rfloor \\
 &\dots \\
 n_k &= \lfloor (p + n_1 + n_2 + \dots + n_{k-1})/s \rfloor
 \end{aligned} \tag{1}$$

We have $n_2 = \lfloor (p + n_1)/s \rfloor$ because the initial page for subchannel 2 is page $p + n_1$. The total number of pages scheduled in one channel is thus $n = \sum_{k=1}^s n_k$. This n is plotted for $p = 100$ in Fig. 1 (the rugged curve), and for $1 \leq p \leq 130$ in Fig. 2. s is varied within the range $1 \leq s \leq p$ (exhaustive search) in both figures. The light-gray curves in Fig. 2 correspond to $p = 9, 21, 51$ and 128. The curve for $p = 9$ 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

⁴In the original paper [7], the parameter is called m . We use q to avoid confusion with other uses of m in this paper.

the curve for $p = 9 + 13 - 1 = 21$. This curve has the peak value of 30, and thus 30 pages can be packed into C_2 , and so forth.

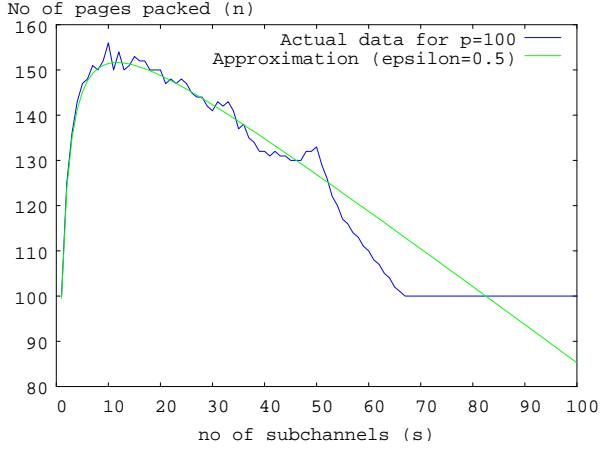


Figure 1: The rugged curve shows the numbers of pages scheduled in one channel when $p = 100$ and the number of subchannels s is varied from 1 to p . The smooth curve is an approximation using Eq. (3) with $\epsilon = 0.5$.

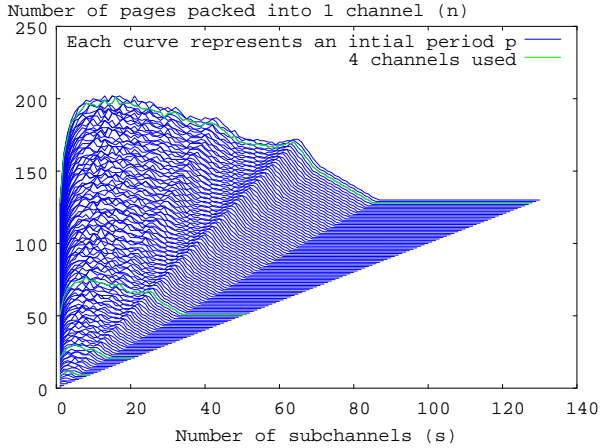


Figure 2: The numbers of pages scheduled in one channel. Each curve corresponds to a different period $p = 1, 2, \dots, 130$ for the first page. The number of subchannels s is varied from 1 to p .

Since it is difficult to manipulate expressions with the floor function, let us approximate the formula for n_k without the floor functions. It is likely that $n_1 = \lfloor p/s \rfloor$, for example, is fairly closer to $p/s - 0.5$ on average. In what follows, we approximate $\lfloor p/s \rfloor$ in the above formulae by $p/s - \epsilon$, where $0 \leq \epsilon < 1$.

$$\begin{aligned} n_1 &\approx p/s - \epsilon = \frac{p - \epsilon s}{s} \\ n_2 &\approx (p + n_1)/s - \epsilon = \frac{p - \epsilon s}{s}(1 + s^{-1}) \\ &\dots \\ n_k &\approx (p + n_1 + n_2 + \dots + n_{k-1})/s - \epsilon \\ &= \frac{p - \epsilon s}{s}(1 + s^{-1})^{k-1} \end{aligned} \quad (2)$$

We thus get

$$\begin{aligned} n &= \sum_{k=1}^s n_k \approx \frac{p - \epsilon s}{s} \sum_{k=1}^s (1 + s^{-1})^{k-1} \\ &= (p - \epsilon s)[(1 + s^{-1})^s - 1] \end{aligned} \quad (3)$$

This n is plotted in Fig. 1 (the smooth curve). It doesn't fit the data curve very well, especially for large s . However, it is seen that the values of s corresponding to the peaks of the exact and approximate curves are fairly close.

We now try to determine the optimal s that maximizes n . Intuitively, s has the following effect on n :

1. A smaller s reduces the number of floor operations that waste bandwidth. (Bandwidth amount $\epsilon - \epsilon$ is wasted on average for each subchannel.)
2. A larger s (more subchannels) can pack more pages, since the initial period is updated (made larger) for each subchannel. \square

To find the optimum s that trades off the above two effects, let us compute $\partial n / \partial s$ and set it to 0, from which we obtain

$$\begin{aligned} (p - \epsilon s)/\epsilon &= \frac{s((1 + s^{-1})^s - 1)}{(1 + s^{-1})^{s-1}((1 + s^{-1})s \log(1 + s^{-1}) - 1)} \\ &\approx \frac{2(e - 1)s^2}{e} \text{ for } s \gg 1 \end{aligned} \quad (4)$$

where \log stands for the natural logarithm. In deriving the above asymptotic limit, we made use of the following two approximations: $(1 + s^{-1})^s \approx e$ for $s \gg 1$, and the Taylor series expansion, $\log(1 + s^{-1}) \approx 0 + s^{-1} - s^{-2}/2 + \dots$.

From Eq. (4), we obtain

$$\frac{ep}{2(e - 1)\epsilon} \approx s^2 + \frac{es}{2(e - 1)} \quad (5)$$

Since the first term on the right hand side dominates the second term for large s , we can ignore the second term in solving Eq. (5) for s :

$$s_{\text{opt}} \approx \sqrt{\frac{ep}{2(e - 1)\epsilon}} \approx \sqrt{0.791p/\epsilon}. \quad (6)$$

If $\epsilon = 0.5$, then

$$s_{\text{opt}} \approx \sqrt{1.58p}.$$

As can be seen from Fig. 3 below, this matches the actual data fairly well. Since the assumption $s \gg 1$ used to derive Eq. (6) is not valid for small s , the bottom figure expands the graph for $1 \leq s \leq 300$.

The middle curve in Fig. 3 represents $s = \sqrt{1.58p}$, and the top and bottom (hardly visible) curves represent $s = \sqrt{2.5p}$ and $s = \sqrt{p}$, respectively. The author of [6] originally thought that $s = \sqrt{p}$ was a good estimate for the optimal

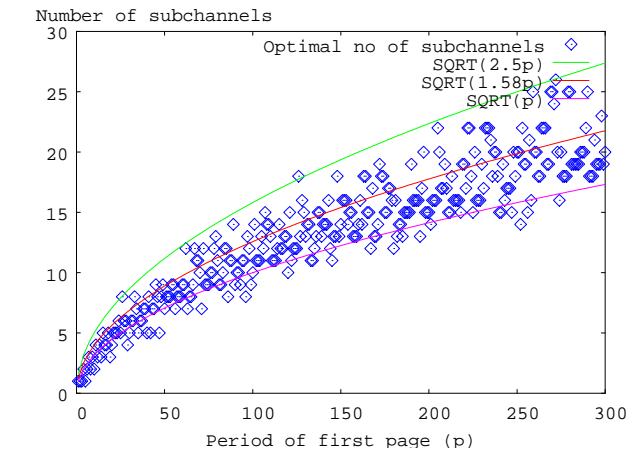
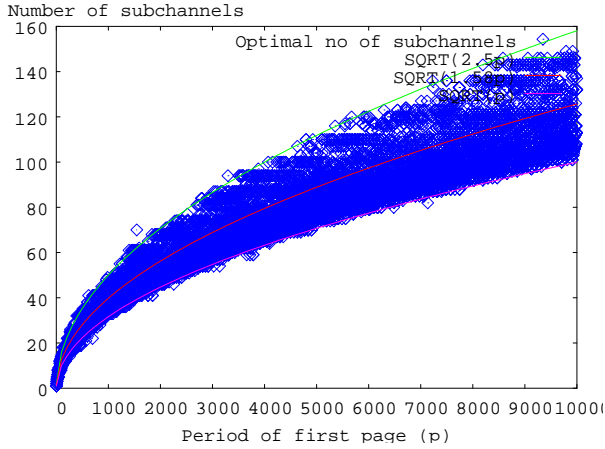


Figure 3: The optimal number of subchannels vs. the period of the first page. The actual optima are plotted as data points.

s value, and later found $\sqrt{p} + 1$ of $\sqrt{p} + 2$ was sometimes optimal [9].

The actual optima were computed by an exhaustive search, varying s from 1 to p to find s_{opt} . From Fig. 3, it would be safe to say that $\max\{\sqrt{p} - 5, 1\} \leq s_{opt} \leq \sqrt{2.5p} + 5$. By reducing the range of search from $s \in [1, p]$ to $s \in [\max\{\sqrt{p} - 5, 1\}, \sqrt{2.5p} + 5]$, the execution time of our search program for all values of p between 1 and 10,000 went down rather drastically from several hours to a few seconds.⁵ Ignoring the “safety fudge factor” of ± 5 , this range is roughly $(\sqrt{2.5} - 1)\sqrt{p} \approx 0.58\sqrt{p}$.

We now plug $s = \sqrt{1.58p}$ into Eq. (3), and plot it in Fig. 4 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. The two curves are hardly distinguishable.

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.

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

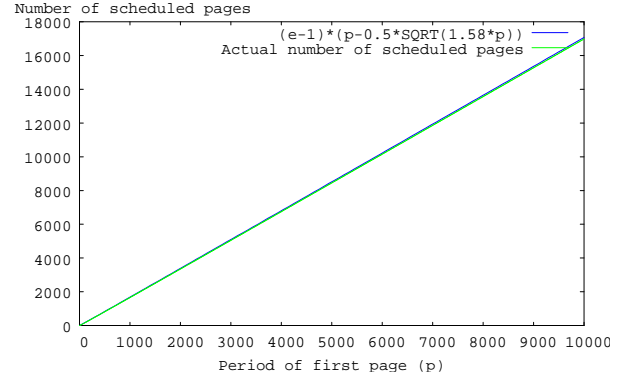


Figure 4: Number of pages scheduled in one channel vs. the period of the first page.

As Fig. 5 shows, we are able to fit slightly more pages into a channel for some values of p . Recursive subchanneling becomes more beneficial for larger values of p . See also the data points indicated by “+” in Fig. 8.

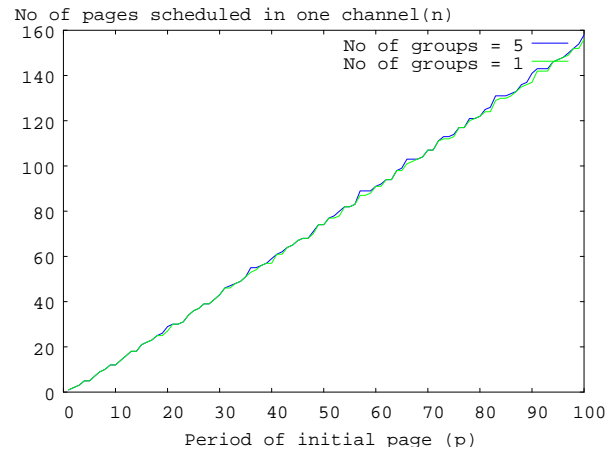


Figure 5: Maximum number of pages scheduled in one channel vs. the period of the first page: The top curve shows the case where up to five groups were considered and best number was chosen, and the bottom curve shows the case where no grouping was used.