Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 154 (2006) 2418–2429
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Generalized Fibonacci broadcasting: An efficient VOD scheme with user bandwidth limit☆

Mingjun Edward Yan, Tsunehiko Kameda*

School of Computing Science, Simon Fraser University, Burnaby, BC, Canada V5A 1S6

Received 30 May 2004; received in revised form 29 October 2004; accepted 21 September 2005
Available online 27 June 2006

Abstract

Broadcasting is attractive in delivering popular videos in video-on-demand service, because the server broadcast bandwidth is independent of the number of users. However, the required server bandwidth does depend on how much bandwidth each user can use, as well as on the user's initial waiting time. This paper addresses the issue of limiting the user bandwidth, and proposes a new broadcasting scheme, named *Generalized Fibonacci Broadcasting* (GFB). In terms of many performance graphs, we show that, for any given combination of the server bandwidth and user bandwidth, GFB can achieve the least waiting time among all the currently known *fixed-delay* broadcasting schemes. Furthermore, it is very easy to implement GFB. We also demonstrate that there is a trade-off between the user waiting time and the buffer requirement at the user.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Multimedia; Video-on-demand; Video broadcasting; Bandwidth minimization

1. Introduction

Video-on-demand (VOD) services aim to deliver videos "instantly" to a large population of users over a high-speed network with broadcast capability. We use the term "user" to mean either a set-top box (STB) on a TV receiver or a computer that can receive a digital video, store it in some storage and concurrently play it from storage at the predefined display rate. The traditional client–server paradigm (so-called "pull technology") is not suitable for delivering "hot" or popular videos to a massive number of users, since it does not scale well. To address the scalability issue, many broadcasting schemes (based on "push technology") have been proposed since 1995, when the pioneering work [17] was published. (A little known work [3] with an almost identical idea was first published as a Philips Research Lab technical report in 1991.) The main idea is to broadcast a set of hot videos periodically, so that a viewer can tune in onto the particular video that s/he wants to watch. Such a scheme typically divides each video into a sequence of segments and broadcasts all the segments periodically on separate (logical) channels. While a viewer is watching the current video segment, it must be guaranteed that the next segment will be downloaded in time for continuous display (continuity condition). Most of the research literature focuses on minimizing the maximum user waiting time for given server bandwidth (or equivalently, minimizing the total server broadcast bandwidth for given maximum waiting time).

☆ This work was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada.
* Corresponding author. Fax: +1 604 291 3045.
E-mail address: tiko@cs.sfu.ca (T. Kameda)
URL: http://www.cs.sfu.ca/∼tiko/ (T. Kameda).

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2006.04.019

The bandwidth here means the sum of the transfer rates (Mbits/s) of the communications channels used concurrently. Clearly, the STB must be able to receive and process data arriving on several concurrent logical channels. The most bandwidth-efficient broadcasting schemes that are currently known require the same bandwidth on the user side as that on server side [5,7,11]. With the current technology, however, the user-side bandwidth (the disk speed, in particular) is likely to be relatively narrow [9], since the STB must be inexpensive. Of course, the server must transmit a number of popular videos concurrently, while a given user receives only one of them at a time. Many broadcasting schemes, such as Pyramid Broadcasting (PB) [17], Permutation-based Pyramid Broadcasting (PPB) [1], Harmonic Broadcasting [7], and Skyscraper Broadcasting (SB) [6], are based on the *fixed start points* policy. With this policy, the STB needs to wait until the beginning of the first segment appears on a channel, before it starts to download the video. More recent results on the performance limits of the broadcasting schemes based on the fixed start points policy can be found in [2,16].

The rest of the broadcasting schemes adopts the *fixed-delay* policy in contrast to the fixed start points policy. In such schemes, the user starts downloading immediately, but waits for a fixed amount of time before s/he starts viewing the video. We propose a new fixed-delay broadcasting scheme, named *Generalized Fibonacci Broadcasting* (GFB), to address the issue of limiting the user-side bandwidth. This scheme is a generalization of a scheme described in a paper by Hu [4]. For any given combination of server bandwidth and user bandwidth, GFB can always achieve the least user waiting time among all the currently known fixed-delay broadcasting schemes. Stated in another way, for any given combination of user bandwidth and waiting time, GFB requires the least server bandwidth.

Another great advantage of GFB is that it uses much fewer segments (hence logical channels) than most other schemes for the same combination of the server bandwidth, user bandwidth and waiting time. Moreover, the mapping from the segments to the logical channels is the simplest possible, i.e., one-to-one. All this implies that it would be very straightforward to implement GFB.

The rest of the paper is organized as follows. Section 2 reviews the currently known broadcasting schemes that take user bandwidth limit into consideration. Section 3 introduces GFB and computes its waiting time as a function of several parameters, such as the server bandwidth, the user bandwidth and the bandwidth of each channel. In Section 4, we will compare GFB with all other similar schemes currently known, and also discuss how its performance is affected as its parameter values are changed. We then derive in Section 5 a formula for computing the buffer size required at the STB. Finally, Section 6 summarizes the contributions of the paper with some remarks. A preliminary version of this paper containing some of the materials presented here appeared in [19,20].

1.1. Basic notation

For simplicity, we concentrate on broadcasting one video. Throughout the paper we use the following notation:

b: video display (or consumption) rate in Mbits/s;
D: total display duration of each video in seconds;
n: number of segments each video is divided into, which equals the number of channels;
S_i: ith segment, $i = 1, 2, \ldots, n$;
D_i: duration of segment S_i in seconds. $\sum_{j=1}^{n} D_j = D$;
d_i: $=D_i/D$;
w: initial waiting time (or latency) in seconds from the time a request is made until display starts;
$w^* = w/D$: normalized wait time; $0 < w^* \leqslant 1$.
B: total server bandwidth required to broadcast one video, expressed as a multiple of b (The server bandwidth will be expressed as either Bb or simply B.);
B_j: bandwidth of the jth broadcast channel as a multiple of b, $j = 1, 2, \ldots, n$; ($B = \sum_{j=1}^{n} B_j$);
U: user bandwidth, expressed as a multiple of b.

2. Previous work

In many existing schemes, the user bandwidth is the same as that the server uses to broadcast a video. Thus, they are not interesting from our perspective (of limiting the user bandwidth) in this paper. Here we review some of the schemes in which the user bandwidth is smaller than the server bandwidth.

PB [17] segments a video in such a way that $d_{i+1} = \alpha d_i$ holds for all $i = 1, 2, \ldots, n-1$ for some constant α, where n is the number of channels. It is shown in [17] that the server bandwidth of PB is given by $B_{\mathrm{PB}} = \alpha n$. The continuity condition for PB only requires that, at any point, the user download from at most two consecutive channels (i.e., channels j and $j+1$). Since all channels have the same bandwidth in PB, the user bandwidth is thus twice the channel bandwidth, i.e.,

$$B_{\mathrm{PB}}^{\mathrm{User}} = 2(B_{\mathrm{PB}}/n) = 2\alpha. \tag{1}$$

The optimal value for α that minimizes B_{PB} is typically larger than 2 [17]. Therefore, if such a value is chosen for α, it follows from Eq. (1) that $B_{\mathrm{PB}}^{\mathrm{User}} > 4$.

PPB [1] is similar to PB, except that each channel is subdivided into p staggered subchannels, and the user uses at most one channel at a time. The user bandwidth requirement in PPB is thus

$$B_{\mathrm{PPB}}^{\mathrm{User}} = B_{\mathrm{PPB}}/(np). \tag{2}$$

If the "optimal" value for α that minimizes B_{PPB} is used, the user bandwidth for PPB (when $p = 2$) is usually greater than 2.5.

SB [6] uses three service processes or threads to receive and playback data segments. As two loader processes download from two streams at the display rate b and fill a buffer with downloaded data, the other thread displays the data. So the user bandwidth requirement of SB is

$$B_{\mathrm{SB}}^{\mathrm{User}} = 2, \tag{3}$$

which is independent of the server broadcast bandwidth.

2.1. Modified schemes to limit user bandwidth

Pâris and Long studied the issue of limiting user bandwidth in [14]. They modified two existing broadcast protocols, namely, Fast Broadcasting (FB) [8] and New Pagoda Broadcasting (NPB) [11], so that the user bandwidth would never exceed three or four times the display rate b. The modified FB with the user bandwidth limited to 3 and 4 will be referred to as FB-3 and FB-4, respectively. Similarly, the modified NPB with the user bandwidth limited to 3 and 4 will be referred to as NPB-3 and NPB-4, respectively. Their study shows that the modification entails a server bandwidth increase by no more than 15%. The server bandwidth requirements of FB-3 and NPB-3 are compared in Fig. 5 in Section 4.

2.2. Fixed-delay schemes

The schemes based on the fixed-delay policy use "greedy" downloading. Namely, the user (STB) starts to download data from the moment it decides to download and display a specific video (this time will be referred to as time 0), and the waiting time (or latency) is always the same, i.e., independent of the time the request is made.

Poly-Harmonic Broadcasting (PHB) [13] was the first scheme to utilize the fixed-delay policy and greedy downloading. To achieve reasonably short waiting time, it must use an excessively large number of channels with increasingly smaller bandwidth. So, this scheme is considered to be of only theoretical interest.

Unlike PHB, both Greedy Equal Bandwidth Broadcasting (GEBB) [5] and Fixed-Delay Pagoda Broadcasting (FDPB) [12] use channels of equal bandwidth. The latter has a parameter m defined by $m = Nw^*$, where w^* denotes the normalized waiting time and N is the number of pages (or fixed-size segments). When parameter m is increased, keeping w^* fixed, its bandwidth requirement decreases, but N must be increased proportionally. The above three schemes all require the user to download data from all channels concurrently, and are the most "efficient" among the fixed-delay schemes in the sense that the asymptotic lower bound on their waiting time is the shortest possible. However, the "costs" incurred in getting close to this bound as their parameters are changed are different for the three schemes. The optimization of the number of subchannels in FDPB is discussed in [10].

FDPB was modified in [12] to limit the user bandwidth. The modified version with the user bandwidth limited to 2 will be referred to as FDPB-2. It is one of the most efficient schemes with a user bandwidth limit.

3. Generalized Fibonacci Broadcasting (GFB)

The *temporal bandwidth map* was introduced in [4] and consists of the playout plane and download plane. Fig. 1 illustrates the temporal bandwidth map of a broadcasting scheme presented in [4]. The playout plane shows a sequence of segments ($n = 5$ segments in this example) that are displayed. Segment S_i is drawn in such a way that its length is proportional to its display duration, Dd_i. The download plane shows that segments S_1 and S_2 are downloaded during the initial waiting time (w), starting at time 0. In general, as soon as the downloading of S_i completes, its display starts, and the downloading of S_{i+2} commences. To satisfy these continuity conditions, the segment durations must be

$$D_1 = 1, \quad D_2 = 2, \quad D_3 = 3, \quad D_4 = 5, \quad D_5 = 8, \quad D_6 = 13,$$

if the duration D_1 of segment S_1 is considered as a unit. They satisfy

$$D_1 = 1, \quad D_2 = 2D_1, \quad \text{and} \quad D_i = D_{i-1} + D_{i-2}, \quad \text{for } i = 3, 4, \ldots .$$

It is observed that these numbers are identical to the Fibonacci sequence (if the first term of the Fibonacci sequence is dropped). We therefore refer to this scheme as Fibonacci Broadcasting. We have $n = 5$ in this example, but other integer values for n will work as well.

The server repeatedly broadcasts each of the n segments on a separate channel with bandwidth b (i.e., normalized bandwidth 1). Therefore, it takes the same amount of time to download a segment as to display it. Note that at most two segments are downloaded concurrently at any time by a user, namely, the user bandwidth is limited to $2b$.

From now on, we will use the normalized duration d_i of each segment S_i over the entire video,

$$d_i = D_i \Bigg/ \sum_{j=1}^{n} D_j = D_i/D. \tag{4}$$

We thus have

$$\sum_{i=1}^{n} d_i = 1. \tag{5}$$

In order to see how we could reduce the waiting time w, note that during the display of the last segment S_5 in the above example, there is no downloading activity. This implies that, during this period, the total user bandwidth equal to $2b$ is being wasted, while during the display period of segment S_4, half of the user bandwidth (equal to b) is being wasted. If we can use channels for a longer time, then we could make better use of them, thus reducing the wasted bandwidth. This can be accomplished by increasing the number of segments (channels), while keeping the total server bandwidth the same, as we see now.

Since it takes (normalized) time d_1 to display S_1 and S_1 must be downloaded in (normalized) time w^* from a channel with bandwidth equal to b, we obviously have $w^* = d_1$. To see the effect of the number of segments n on the waiting time w^*, let us compute w^* for Fibonacci Broadcasting with different values of n. For a given value of n in column 1, column 2 of the table in Fig. 2 shows d_n as a multiple of d_1.

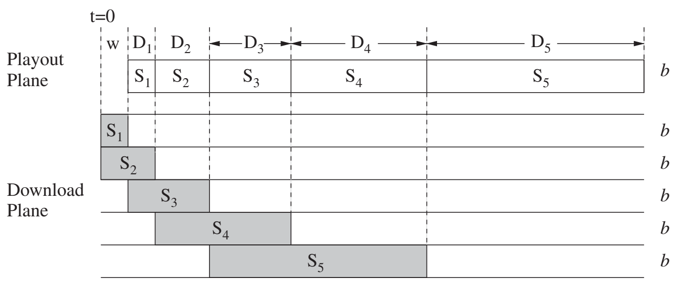

Fig. 1. Temporal bandwidth map of Fibonacci Broadcasting.

Server bandwidth (n)	Segment length	Waiting time ($w^* = d_1$)
2	$d_2 = 2d_1$	0.3333
3	$d_3 = 3d_1$	0.1667
4	$d_4 = 5d_1$	0.09091
5	$d_5 = 8d_1$	0.05263
6	$d_6 = 13d_1$	0.03125
7	$d_7 = 21d_1$	0.01887
8	$d_8 = 34d_1$	0.01149
9	$d_9 = 55d_1$	0.007042
10	$d_{10} = 89d_1$	0.004329
11	$d_{11} = 144d_1$	0.002667
12	$d_{12} = 233d_1$	0.001645

Fig. 2. Waiting time vs. server bandwidth in Fibonacci Broadcasting.

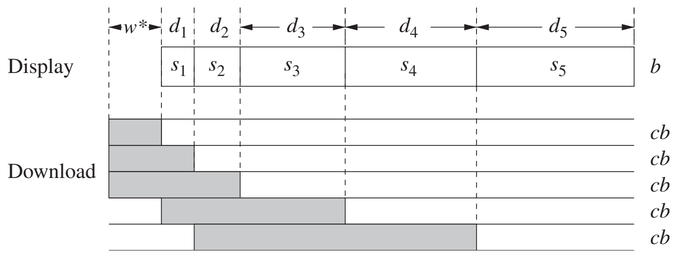

Fig. 3. Illustration of Generalized Fibonacci Broadcasting (GFB): $K = 3$ and $c = 1/g$ in this illustration.

For each value of n in column 1, column 3 shows the normalized waiting time. The entry at row 2 ($n = 3$) and column 3, for example, was computed as follows. Since $d_3 = 3d_1$ and $d_2 = 2d_1$, we have $1 = d_3 + d_2 + d_1 = 6d_1$, hence $d_1 = \frac{1}{6}$ and $w^* = d_1 \approx 0.1667$. Column 3 shows that increasing n is very beneficial for reducing the waiting time.

Unfortunately, increasing n by 1 entails the addition of one more channel of bandwidth b. In this paper, we are interested in minimizing the waiting time w (or w^*) for fixed server bandwidth and user bandwidth. Therefore, if we double n, for example, we must cut the bandwidth of each channel by half, so that the total bandwidth requirement remains the same. With this introduction, we can now present our model. Just like Fibonacci Broadcasting, on the server side, each video is divided into n segments and each segment S_i is broadcast on its dedicated logical channel, so that the number of channels equals n. Unlike Fibonacci Broadcasting, however, each channel now has the same bandwidth b/g, where g (> 0) is a configurable parameter. At time 0, the user begins to download video segments from the first K (instead of 2) channels, where K is an integer constant satisfying $1 \leqslant K \leqslant n$. After fixed delay, w ($= Dw^*$), the user can begin to display S_1.

As shown in Fig. 3, at this time, downloading from the first channel stops, as the user moves on to download from channel $K + 1$, while continuing to download data from channels $2, \ldots, K$. In general, for $i = 1, 2, \ldots, n - 1$, the user ceases to receive segment S_{i+1} when the previous data segment S_i has been completely displayed. Since we have $B_i = 1/g$ for all i by definition, the server bandwidth is simply

$$B_{\mathrm{GFB}} = n/g \tag{6}$$

and the user bandwidth is limited to K/g, or

$$B_{\mathrm{GFB}}^{\mathrm{User}} = K/g. \tag{7}$$

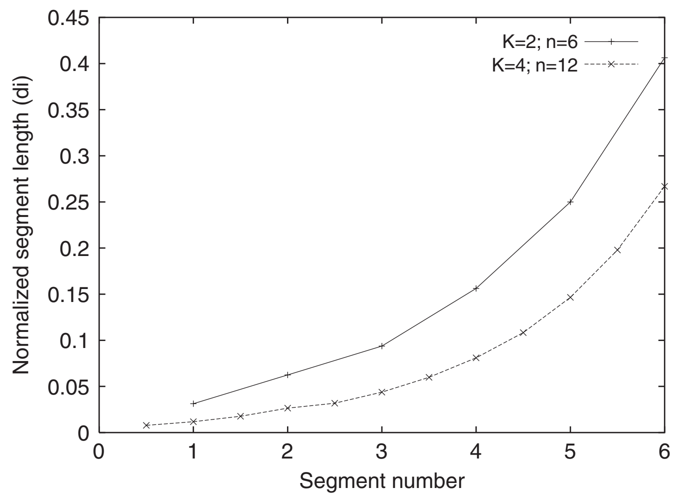

Fig. 4. Normalized segment lengths $\{d_i\}$ for the cases ($K=2$, $g=1$, $n=6$) and ($K=4$, $g=2$, $n=12$).

From Fig. 3, we can derive the continuity conditions of GFB as follows:

$$
\begin{aligned}
&d_1 = (1/g)w^*, &(8)\\
&d_2 = (1/g)(w^* + d_1) = [1 + (1/g)]d_1,\\
&d_3 = (1/g)(w^* + d_1 + d_2) = [1 + (1/g)]^2 d_1,\\
&\quad \ldots\\
&d_K = (1/g)(w^* + d_1 + \cdots + d_{K-1}) = [1 + (1/g)]^{K-1} d_1, &(9)\\
&d_{K+1} = (1/g)(d_1 + \cdots + d_K) = ([1 + (1/g)]^K - 1)d_1, &(10)\\
&d_{K+2} = (1/g)(d_2 + d_3 + \cdots + d_{K+1}),\\
&d_{K+3} = (1/g)(d_3 + d_4 + \cdots + d_{K+2}),\\
&\quad \ldots\\
&d_n = (1/g)(d_{n-K} + d_{n-K+1} + \cdots + d_{n-1}).
\end{aligned}
$$

The above recurrence relation resembles the generalized Fibonacci sequence. Hence the name GFB. We use GFB(K/g) to denote GFB with parameters K and g. The total number of channels, n, is another design parameter, which is not explicitly specified, since we want to vary it for a fixed pair of K and g values. If we set $K=2$ and $g=1$, for example, then each broadcast channel has normalized bandwidth equal to $1/g=1$, and the user downloads data from at most $K=2$ concurrent channels, limiting the user bandwidth to $U=K/g=2$. As in Fig. 2, different choices for n lead to different waiting time w^*.

By substituting Eq. (8) into the equation below it, we can express d_2 in the form $d_2 = f_2(g)d_1$, where $f_2(g)$ is a function of g. Similarly, for each $i=2,\ldots,n$, we can express d_i in the form $d_i = f_i(g,K)d_1$, where $f_i(g,K)$ is a function of g and K. Finally, by substituting all d_i into Eq. (5), we get

$$d_1 = \frac{1}{1 + \sum_{i=2}^{n} f_i(g,K)}. \tag{11}$$

Since $w^* = gd_1$ from Eq. (8), we now can express w^* in terms of g and K. Note that d_1 of Eq. (11), hence w^*, implicitly depends on n. Given g and K, for different values of n, one can compute $B=n/g$ and w^*.

Fibonacci Broadcasting is clearly a special case of GFB, where $g=1$. In Fig. 4, $\{d_i\}$ for two different combinations of parameter values are plotted. The +'s represent $d_1, d_2, \ldots, d_6$, when $K=2$, $g=1$, and $n=6$, which can be computed from the above recurrence relation (they are also given in Fig. 2). Similarly, the ×'s at 0.5, 1, 1.5, …, 5.5, and 6 (on the horizontal axis) represent $d_1, d_2, \ldots, d_{12}$, when $K=4$, $g=2$, and $n=12$. Thus, in these two cases, the user bandwidth is the same since $K/g = 2$. It is seen from Fig. 4 that d_1 for the latter case is much smaller than that of the former. Therefore, increasing the number of channels from 6 to 12 has a beneficial effect of reducing the waiting time $w^* = gd_1$, since g doubled but d_1 was reduced to much less than half.

4. Comparisons

In Section 4.1, we compare several known schemes with fixed user bandwidth and GFB(K/g). We first fix the user bandwidth to 3 or 2, and compare the server bandwidth and/or waiting time (w) of different schemes.

4.1. GFB vs. other broadcasting schemes with user bandwidth limit

Fig. 5 compares the waiting time of FB-3 [14] and NPB-3 [14] with that of GFB(9/3), each of which limits the user bandwidth to 3. It is seen that FB-3 is the clear loser among the three. Actually, the relationship between the waiting time w and the server bandwidth B for FB-3 is the same as that for GFB(3/1) shown in Fig. 9. However, FB-3 uses much more segments (hence channels).

Fig. 6 compares, FDPB-2 (with its parameter $m = 100$) [12], GFB(6/3) and GFB(8/4), all of which limit the user bandwidth to 2. (The waiting time achieved by FDPB-2 for the server bandwidth above 7 are not available to us.) For example, for server bandwidth of 7, GFB(6/3) and FDPB-2 achieve waiting times of 42.7 and 49.3 s, respectively, for a 120 min video. GFB(6/3) achieves this using $n = 7g = 21$ segments (i.e., 21 logical channels). Each “channel” (of bandwidth equal to the display rate b) of FDPB-2, however, is divided into many “subchannels” (e.g., 7 “channels” are divided into 458 “subchannels”). “Subchannels” are really separate logical channels, and as such require demultiplexing within a “channel” at the user, which incurs additional processing time. It is also observed that, with server broadcasting bandwidth of 4 and 5, FDPB-2 achieves shorter waiting times than GFB(6/3). A nice feature of GFB(K/g), however, is that there are so many ways to choose the parameters K and g, while keeping the user bandwidth $U = K/g$ fixed. For example, Fig. 6 also plots GFB(8/4), which has shorter waiting time than both FDPB-2 and GFB(6/3). FDPB-2 also has a parameter m that can be increased to improve its performance. However, it will increase the number of “subchannels” even further.

4.2. Choosing values for K and g

In Fig. 7, we vary parameters K and g, while keeping user bandwidth $U = K/g$ fixed at 2. Note that the waiting time w^* becomes shorter, as K is increased.

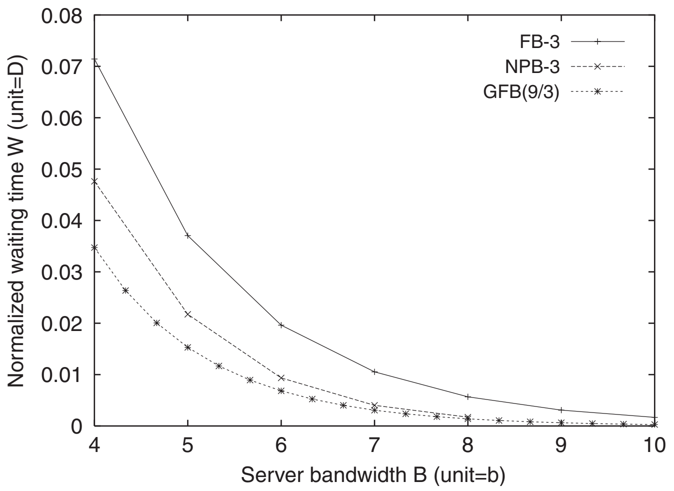

Fig. 5. Normalized waiting time vs. server bandwidth B for FB-3, NPB-3, and GFB(9/3), from top to bottom. (In this and subsequent figures, normalized wait time w^* is shown as W.)

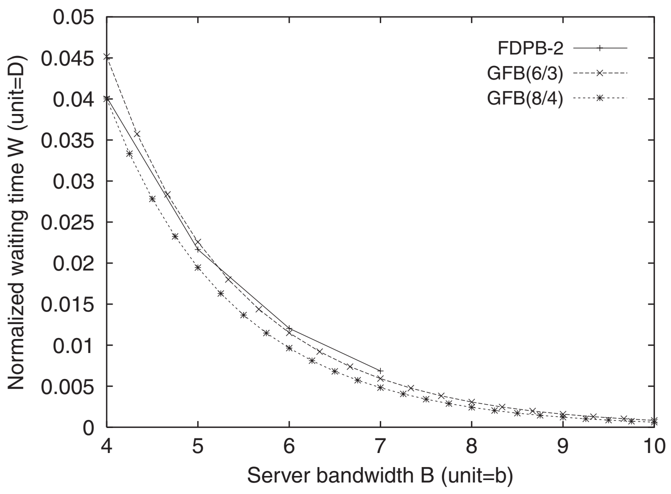

Fig. 6. Waiting time vs. server bandwidth for FDPB-2 ($m = 100$), GFB(6/3), and GFB(8/4), from top to bottom.

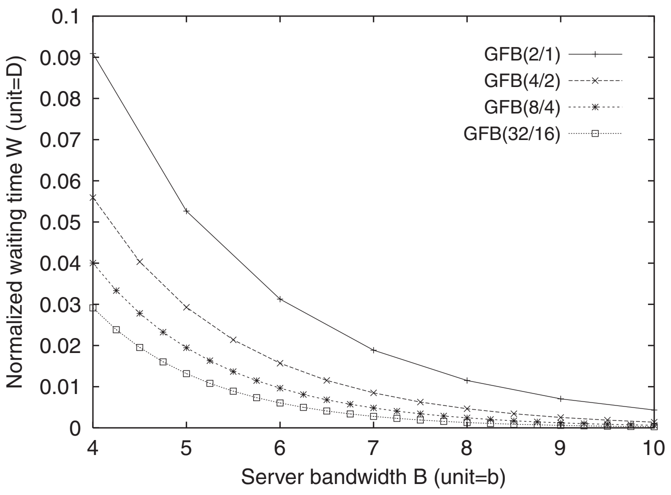

Fig. 7. Waiting time vs. server bandwidth for GFB(K/g) with $K/g = 2$: GFB(2/1), GFB(4/2), GFB(8/4), and GFB(32/16), from top to bottom.

Fig. 8 is a similar graph with $K/g = 4$. The first five curves from the top are for $K = 5$, 6, 7, 8 and 9, respectively. The bottom two curves are for $K = 20$ and 21, and they are hardly distinguishable.

From Figs. 7 and 8, we can see that if we keep the user bandwidth $U = K/g$ constant, increasing K and g proportionally leads to reduced server bandwidth for GFB(K/g). Note that K is an integer, but g can be an arbitrary positive number. In general, there is not much gain when g $(=K/U)$ is increased beyond 4 (for $U \geqslant 4$; larger for $U = 2$). For the bottom two overlapping curves, $g \geqslant 10$ holds.

To see the relative effect of changing K with $U = K/g$ fixed vs. changing U, Fig. 9 plots two curves, GFB(3/1) and GFB(6/2), together with GFB(2/1) and GFB(4/2). An interesting phenomenon is the relative performance of GFB(4/2) vs. GFB(3/1). Even though GFB(4/2) limits the user bandwidth to 2, while GFB(3/1) limits it to 3, GFB(4/2) performs better than GFB(3/1). This implies that increasing the number of user channels K from 3 to 4 more than compensates for the reduction of user bandwidth from 3 to 2. However, if we compute the values of $g = K/U$ for these two cases, we get $g = 1$ for GFB(3/1) and $g = 2$ for GFB(4/2). Based on this and other evidences, we believe that g is more important than K in determining the performance of GFB(K/g) when U varies.

Finally, Fig. 10, shows the performance of GFB(K/g), when K is varied while keeping g constant ($g = 2$ in this case). From this figure, we can see how quickly the performance improves as K is increased. In the extreme case where

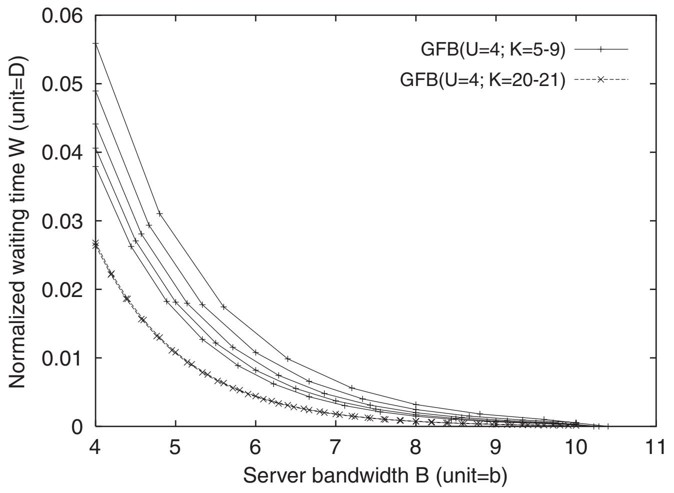

Fig. 8. Waiting time vs. server bandwidth for GFB(K/g) with $K/g = 4$: GFB(5/(5/4)), GFB(6/(3/2)), GFB(7/(7/4)), GFB(8/2), GFB(9/(9/4)), GFB(20/5), and GFB(21/(21/4)), from top to bottom.

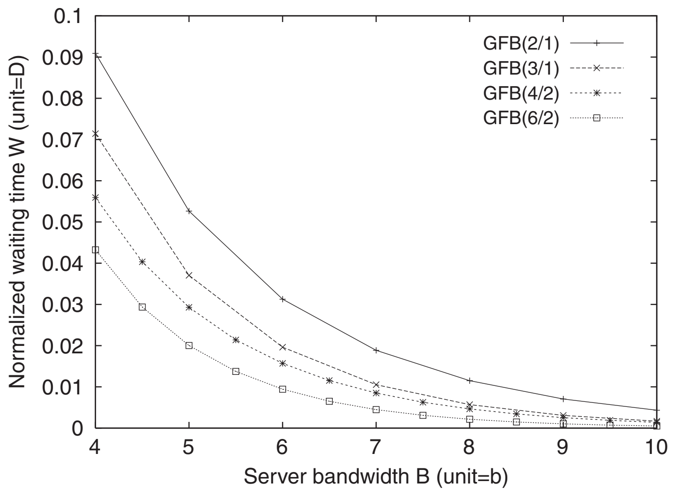

Fig. 9. Comparison of GFB(2/1), GFB(3/1), GFB(4/2), and GFB(6/2).

the user downloads from all available broadcast channels concurrently ($K = n$), GFB coincides with Greedy Equal Bandwidth Broadcasting [5] (see Section 2.2).

4.3. Buffer requirement

Let t_j denote the time (measured from the display start time) when segment S_j has just been downloaded and is about to be displayed. It is easy to see that t_j can be calculated as

$$t_j = \sum_{i=1}^{j-1} D_i = D \sum_{i=1}^{j-1} d_i.$$

Clearly, at time t_j. the buffer must contain all of segment S_j. In addition, parts of segments $S_{j+1}, S_{j+2}, \ldots, S_{j+K-1}$ are also in the buffer, since up to K segments are concurrently downloaded. Let $m = j + K - 1$. Since the downloading

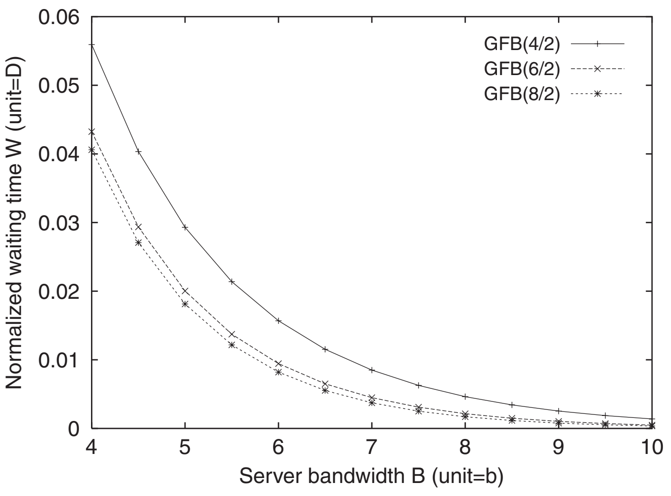

Fig. 10. Waiting time vs. server bandwidth for GFB($K/2$): GFB(4/2), GFB(6/2), and GFB(8/2), from top to bottom.

of S_m completes at time t_m, after the display start time of segment S_j it will take time $t_m - t_j$ until the downloading of S_m will complete. Let us use the notation

$$T(j,m) = t_m - t_j = D \sum_{i=j}^{m-1} d_i.$$

Since each channel has bandwidth b/g, the total download time T_m for segment S_m (of size $Dd_m b$) is given by

$$T_m = Dd_m g.$$

The amount of data in segment S_m is $Dd_m b$. Therefore, the amount of data $P_j(m)$ from segment S_m that is in the buffer at time t_j is

$$P_j(m) = Dd_m b\left(\frac{T_m - T(j,m)}{T_m}\right) = Db\left(d_m - \frac{1}{g}\sum_{i=j}^{m-1} d_i\right).$$

Define $u(j) = \min\{n, j+K-1\}$ as the maximum segment number that is buffered at time t_j. The amount of data in the buffer at time t_j is given by

$$\begin{aligned} Dd_j b + \sum_{m=j+1}^{u(j)} P_j(m) &= Db\left[d_j + \sum_{m=j+1}^{u(j)}\left(d_m - \frac{1}{g}\sum_{i=j}^{m-1} d_i\right)\right] \\ &= Db\left[\sum_{m=j}^{u(j)} d_m - \frac{1}{g}\sum_{m=j+1}^{u(j)}\sum_{i=j}^{m-1} d_i\right] \\ &= Db\left[\sum_{m=j}^{u(j)} d_m - \frac{1}{g}\sum_{i=j}^{u(j)-1}(u(j)-i)d_i\right]. \end{aligned}$$

Note that the above buffer size depends on g, K, n and d_i. Quantity d_j in turn is a function of K, g and n.

Fig. 11 plots the buffer size at the points in time when a segment has been completely downloaded and is about to be displayed. The server bandwidth is set to 5, so that $B = n/g = 5$, and each user can download from up to $K = 4$ channels. We vary the parameter g in GFB($4/g$) from 1 to 4. For GFB(4/1), for example, we get $n = 5$ from $n/g = 5$. In other words, the video is divided into five segments, and each segment is broadcast on a channel with bandwidth

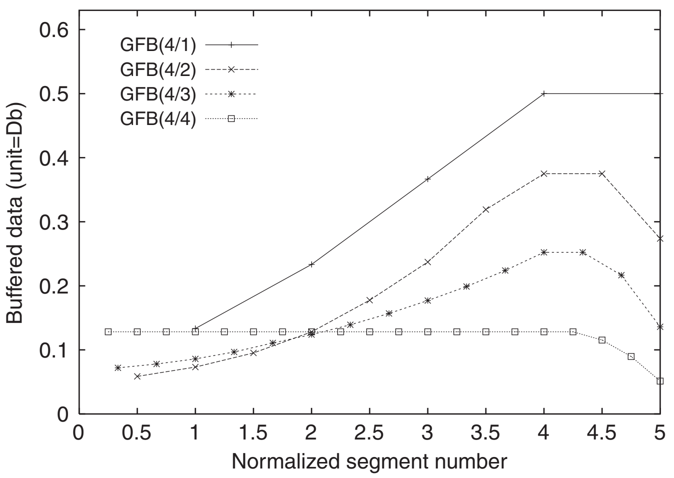

Fig. 11. Buffer size comparison ($B = 5$).

equal to the display rate b. Each of the five data points on the solid curve in Fig. 11 shows the amount of data in the buffer at the time when the corresponding segment has just been downloaded in its entirety. Initially segments S_1 to S_4 are downloaded from channels 1 to 4. Once S_1 has been completely downloaded and its display starts, segment S_5 starts to be downloaded on channel 5. From this time on, the buffer is filled at the rate of $4b$, while data are taken out of the buffer for display at the rate of b. Therefore, the buffer grows at the net rate of $3b$. As the display of S_1 finishes, so does the downloading of S_2. From then on, only segments S_3 to S_5 will enter the buffer at a total rate of $3b$ (net rate is $2b$). Eventually, while S_4 is being displayed only S_5 is downloaded, so that the net rate into the buffer becomes 0. During this time, the buffer size stays the same, as seen in Fig. 11. Let us now consider GFB(4/4). Since $K = 4$ and $g = 4$ in this case, we have $U = K/g = 1$, i.e., the user's bandwidth is only b. Since the downloading rate is the same as the display rate, the buffer size stays constant as seen in the bottom curve in Fig. 11. Towards the end, however, less than four channels of bandwidth $b/4$ are used, and therefore, the buffer size starts to decrease. Since $n/g = 5$, we have $n = 20$. This is why there are 20 data points in the bottom curve. In Fig. 11, the segment numbers are "normalized", so that segment S_i is plotted at horizontal position i/g.

It can be seen that the maximum buffer size (peak of each curve) is larger for larger U. In the worst case (GFB(4/1)), as much as 50% of the video is in the buffer. However, there is a trade-off between the buffer size and the waiting time, and the waiting time is smaller for larger U, as we observed in Fig. 10.

5. Summary and conclusion

We have proposed a new broadcasting scheme, called Generalized Fibonacci Broadcasting (GFB). It has three configurable parameters, n, g and K, where $g > 0$ and K is an integer such that $1 \leqslant K \leqslant n$. In GFB($K/g$), the server uses n channels, broadcasting each video segment on a dedicated channel with bandwidth b/g, where b (Mbits/s) is the video display rate. The user downloads data from at most K out of the n channels. Therefore, the user bandwidth is limited to Kb/g, and the server bandwidth is nb/g. This gives a service provider great flexibility in choosing the appropriate parameter values, according to the technology (and the budget) available in the given environment, such as the bandwidths available to the users as well as the server. We demonstrated that increasing K and g proportionally, while keeping user bandwidth Kb/g fixed, has the effect of reducing the server bandwidth required to achieve a given waiting time (or reducing the waiting time for a given server bandwidth). Choosing a value $g < 1$ is possible, but the performance degrades.

Another great advantage of GFB is that it will be straightforward to implement it. For example, GFB(6/3) with $n = 21$ broadcasting channels (hence 21 segments) can achieve a shorter waiting time (about 0.6% of the total video length) than FDPB-2 (with parameter $m = 100$), which is one of the most efficient broadcasting schemes with user bandwidth limit that are currently known. In actual implementation, it may be desirable to choose an integer as g. Then an existing channel with bandwidth b could be time-division multiplexed into g logical subchannels relatively easily. Last but not least, GFB gives fine-grained choices for the server and user bandwidths, provided by different combinations of K and g. It is clear that GFB can be combined easily with the idea of “aggressive preloading” [15] (prefix caching) to reduce the waiting time and/or the server bandwidth. The only change required is to set $w^* = d_0$ in Eqs. (8) and (9) and to change Eq. (5) to $\sum_{i=1}^{n} d_i = 1$, where d_0 is the normalized duration of the cached prefix. Recently, William and Kirkpatrick derived an asymptotic lower bound for the server bandwidth of GFB [18].

Acknowledgments

The authors would like to thank Bill Cressman of Simon Fraser University for contributing the idea that formed the basis for the analysis of Section 4.

References

[1] C. Aggarwal, J. Wolf, P. Yu, A permutation based pyramid broadcasting scheme for video-on-demand, in: Proceedings of the International Conference on Measurement and Modeling of Computer Systems, IEEE, Philadelphia, 1996, pp. 200–209.

[2] A. Bar-Noy, R. Ladner, T. Tamir, Scheduling techniques for media-on-demand, in: Proceedings of the Symposium on Discrete Algorithms, ACM–SIAM, 2003, pp. 791–800.

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

[4] A. Hu, Video-on-demand broadcasting protocols: a comprehensive study, in: Proceedings of the INFOCOM, IEEE, Anchorage, AK, 2001, pp. 508–517.

[5] A. Hu, I. Nikolaidis, P. Beek, On the design of efficient video-on-demand broadcast schedules, in: Proceedings of the Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MAS-COTS), 1999, pp. 262–269.

[6] K. Hua, S. Sheu, Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems, in: Proceedings of the SIGCOMM, ACM, Cannes, France, 1997, pp. 89–100.

[7] L. Juhn, L. Tseng, Harmonic broadcasting for video-on-demand service, IEEE Trans. Broadcasting 43 (1997) 268–271.

[8] L. Juhn, L. Tseng, Fast broadcasting for hot video access, in: Proceedings of the Fourth International Workshop on Real-Time Computing Systems and Applications, 1997, pp. 237–243.

[9] L. Juhn, L. Tseng, Fast data broadcasting and receiving scheme for popular video service, IEEE Trans. Broadcasting 44 (1998) 100–105.

[10] T. Kameda, Y. Sun, L. Goddyn, An optimization problem related to vod broadcasting, in: Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC), 2005, pp. 116–125.

[11] J.-F. Pâris, A simple low-bandwidth broadcasting protocol for video-on-demand, in: Proceedings of the Eighth International Conference on Computer Communications and Networks (IC3N), Boston, Natick, MA, 1999, pp. 118–123.

[12] J.-F. Pâris, A fixed-delay broadcasting protocol for video-on-demand, in: Proceedings of the 10th International Conference on Computer Communications and Networks, 2001, pp. 418–423.

[13] J.-F. Pâris, S.W. Carter, D. Long, A low bandwidth broadcasting protocol for video on demand, in: Proceedings of the International Conference on Computer Communications and Networks (IC3N), IEEE, 1998, pp. 690–697.

[14] J.-F. Pâris, D. Long, Limiting the user bandwidth of broadcasting protocols for video-on-demand, in: Proceedings of the Euromedia Conference, Antwerp, Belgium, 2000, pp. 107–111.

[15] J.-F. Pâris, D. Long, The case for aggressive partial preloading in broadcasting protocols for video-on-demand, in: Proceedings of the International Conference on Multimedia and (ICME), IEEE, Tokyo, 2001, pp. 113–116.

[16] Y. Sun, T. Kameda, Harmonic block windows scheduling through harmonic windows scheduling, in: Proceedings of the 11th International Workshop on Multimedia Information Systems, Sorrento, Italy, Lecture Notes in Computer Science, vol. 3665, Springer, Berlin, 2005, pp. 190–206.

[17] S. Viswanathan, T. Imielinski, Pyramid broadcasting for video on demand service, in: Proceedings of the Multimedia Computing and Networking Conference, vol. 2417, IEEE, San Jose, CA, 1995, pp. 66–77.

[18] E. William, D. Kirkpatrick, Optimally scheduling video-on-demand to minimize delay when server and receiver bandwidth may differ, in: Proceedings of the Symposium on Discrete Algorithms (SODA), ACM–SIAM, 2004, pp. 1041–1049.

[19] E. Yan, Minimizing bandwidth requirement of broadcasting protocol in video-on-demand services, M.Sc. Thesis, School of Computing Science, Simon Fraser University, April 2002.

[20] E. Yan, T. Kameda, An efficient vod broadcasting scheme with user bandwidth limit, in: Proceedings of the 10th Multimedia Computing and Networking (MMCN), SPIE–ACM, 2003, pp. 200–208.