

An Efficient VOD Broadcasting Scheme with User Bandwidth Limit

Edward Mingjun Yan and Tiko Kameda
School of Computing Science, Simon Fraser University
Burnaby, British Columbia, Canada, V5A 1S6
{eyan, tiko}@cs.sfu.ca

Abstract

*To address the scalability issue in **video-on-demand** systems, many broadcasting schemes have been proposed to date. The major performance parameters of such a broadcasting scheme are the server broadcast bandwidth, the user bandwidth and the user's initial waiting time. The broadcasting schemes with the least server bandwidth requirement currently known require the same bandwidth on the user side as that on the server side. We propose a new broadcast scheme, named **Generalized Fibonacci Broadcasting (GFB)**, to address the issue of limiting the user-side bandwidth requirement. For any given combination of the server and user bandwidths, GFB can always achieve the least user waiting time among all the currently known broadcasting schemes. Furthermore, it would be very easy to implement GFB.*

1. Introduction

Video-on-demand (VOD) services aim to provide videos to a large population of users over a high-speed network with broadcast capability. The "user" is either a **set-top box (STB)** or a computer with capabilities of receiving a digital video, storing it in secondary storage and concurrently playing it from storage at the predefined **display rate**. The traditional client-server paradigm (so-called "pull technology") is not suitable for VOD systems, since it does not scale well to a massive number of potential users. To address the scalability issue, many broadcasting schemes (based on "push technology") have been proposed since 1995, when the pioneering work [18] was published. The main idea is to broadcast a set of high-demand (or "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 each segment periodically on a separate logical channel, concurrently with other segments. 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 a given server bandwidth (or equivalently, minimizing the total server broadcast bandwidth for a given maximum waiting time required for the users). The **bandwidth** here means the sum of the transfer rates (Mbits/sec) of the communications channels used concurrently, which implies a certain processing speed on the part of the sender/receiver processors. In particular, the user must be able to receive and process data arriving on several concurrent logical channels. The best broadcasting schemes in this sense that are currently known require the same bandwidth on the user side as that on server side [4,10,11,12]. With the current technology, however, the user-side bandwidth requirement is likely to be the bottleneck [8,16], since the user equipment must be inexpensive. Of course, the server must m videos concurrently, while a given user receives only one of them at a time, where m is the number of popular videos.

We propose a new broadcast scheme, named **Generalized Fibonacci Broadcasting (GFB)**, to address the issue of limiting the user-side bandwidth requirement. This scheme is a generalization of a scheme described in a paper by A. Hu [3]. For any given combination of server and user bandwidths, GFB can always achieve the least user waiting time among all the currently known 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 channels) than others 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.

Section 2 reviews the currently known broadcasting schemes for limiting the user bandwidth. In Section 3, we introduce GFB. In Section 4, we will compare GFB with all other similar schemes currently known, and also discuss its behavior as its parameters are changed. Finally, Section 5 summarizes the contributions of the paper with some remarks.

Basic Notation

For simplicity, we assume in this paper that all videos have the same length (size and duration), and concentrate on one such video. Throughout the paper we use the following notation:

- b - video **display** (or **consumption**) **rate** in Mbits/sec;
- D - total display **duration** of each video in seconds;
- S_i - i th segment, $i=1,2, \dots, n$;
- n - number of segments each video is divided into;
- d_i - duration of segment S_i in seconds;
- w - **waiting time** (= **access time**) in seconds, i.e., the maximum time experienced by the viewer from the time a request is made until display starts;

$$W = w/D, \text{ normalized waiting time; } 0 < W \leq 1.$$

B - total **server bandwidth** required to broadcast one video, expressed as a (dimensionless) ratio over b ; The server bandwidth will be expressed as either $B*b$ or simply B .

B_j - bandwidth of the j th broadcast channel as a ratio over b , $j=1,2, \dots, N$;

N - number of logical broadcast channels;

U - user bandwidth, expressed as a ratio over b ;

The following equalities follow directly from definition:

$$D = \sum_{i=1}^n d_i \quad (1.1)$$

$$B = \sum_{j=1}^N B_j \quad (1.2)$$

2. Previous work

In earlier broadcasting schemes, the STB needed to wait until the beginning of the first segment is broadcast, before it started to download the video. They include *Pyramid Broadcasting (PB)* [18, 19], *Permutation-based Pyramid Broadcasting (PPB)* [1, 2], *Skyscraper Broadcasting (SB)* [5], *Enhanced Harmonic Broadcasting* [10], *Staircase Broadcasting* [8], *Harmonic Broadcasting* [6], *Cautious Harmonic Broadcasting* [13], *Quasi-Harmonic Broadcasting* [13],

and *Pagoda Broadcasting* [14]. Except for PB, PPB, and SB, the user bandwidth requirements of all these broadcasting schemes are the same as their server broadcast bandwidth requirements. Thus, they are not interesting from our perspective (of limiting the user bandwidth) in this paper, even though some of them achieve relatively small user bandwidth.

The performance of those earlier schemes has been exceeded by more recent “greedy” schemes, and therefore, we will not go into details of those, but simply mention some results which are relevant to our discussion.

PB was the first novel broadcasting scheme, which stimulated a great deal of research interest in VOD broadcasting. PB segments a video in such a way that $d_{i+1} = \alpha*d_i$ holds for all $i = 1, 2, \dots, n-1$ for some constant α . It is shown in [18, 19] that the server bandwidth of PB is given by $B_{PB} = \alpha*N$. 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.,

$$B_{PB}^{User} = 2*B_{PB}/N = 2*\alpha. \quad (2.1)$$

The optimal value for α that minimizes B_{PB} is typically larger than 2. Therefore, if such a value is chosen for α , it follows from (2.1) that $B_{PB}^{User} \geq 4$.

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

$$B_{PPB}^{User} = B_{PPB} / (N*p). \quad (2.2)$$

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

SB 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

$$B_{SB}^{User} = 2, \quad (2.3)$$

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 [16]. They modified two existing broadcast protocols, namely, *Fast Broadcasting (FB)* [7, 9] and *New Pagoda Broadcasting (NPB)* [15], so that the user bandwidth would never exceed three or four times the

display rate b . Since each “channel” used in FB and NPB has bandwidth b , in the modified FB and NPB, the user accesses three or four “channels” at the same time. Their study shows that the modification entails a server bandwidth increase by no more than 15%.

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. The server bandwidth requirements of FB-3 and NPB-3 are compared in Fig. 5 in Section 4.

2.2. Fixed-delay schemes

The three most efficient schemes use “greedy” downloading and fixed delay. Namely, the user (STB) starts to download data from the moment it decides to display a specific video (this time will be referred to as **time 0**), and the waiting time (=delay) is always the same, i.e., independent of the time the request is made. They all require the user to download data from all channels concurrently, and are the most “efficient” in the sense that the asymptotic lower bounds on their waiting times are given by formula (4.1). However, the “costs” incurred in getting close to this bound as their parameters are changed are different for the three schemes.

Poly-Harmonic Broadcasting (PHB) [12] was the first scheme to utilize greedy downloading. To achieve a reasonably short waiting time, it must use an excessively large number of channels with increasingly smaller bandwidths. So, this scheme is considered to be of only theoretical interest.

Unlike PHB, both *Greedy Equal Bandwidth Broadcasting (GEBB)* [4] and *Fixed-Delay Pagoda Broadcasting (FDPB)* [Pâr01] use channels of equal bandwidth. The latter has a parameter m , which satisfies $m = n * W$. When parameter m is increased, keeping W fixed, then its bandwidth requirement decreases, but n must be increased proportionally, often causing each video to be partitioned into a very large number of small segments. [Pâr01] modified FDPB to limit its 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 [3] and consists of the **playout plane** and **download plane**.

The upper half of Fig.1 illustrates the temporal bandwidth map of a certain broadcasting scheme presented in [3].

The playout plane shows a sequence of segments ($n=6$ 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, d_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} also commences. To satisfy these continuity conditions, the segment durations must be

$$d_1=1, d_2=2, d_3=3, d_4=5, d_5=8, d_6=13,$$

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

$$d_1=1,$$

$$d_2=2*d_1,$$

$$d_i = d_{i-1} + d_{i-2}, \text{ for } i = 3, 4, \dots$$

It is observed that these numbers are identical to the *Fibonacci* sequence (if the first term of the *Fibonacci* sequence is removed). We therefore refer to this scheme as **Fibonacci Broadcasting**. In this scheme N (number of channels) = n (number of segments) always holds and $n = 6$ in this example, but, other integer values for $N=n$ will work as well.

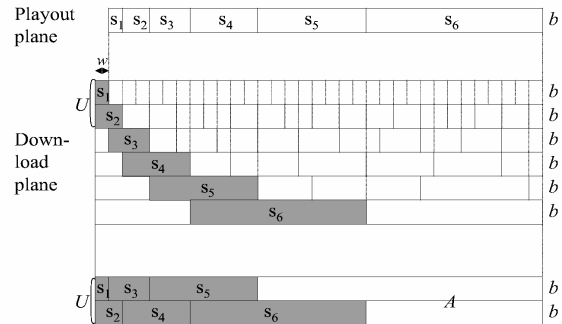


Figure 1. Temporal bandwidth map of “Fibonacci Broadcasting”.

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 $2b$ ($U=2$ in the figure).

From now on, we will mostly use the normalized duration (or length) of each segment S_i over the entire video,

$$L_i = d_i / D. \quad (3.1)$$

We thus have from (1.1):

$$\sum_{i=1}^n L_i = 1. \quad (3.2)$$

In order to see how we could reduce the waiting time w , note that during the display of the last segment S_6 , 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_5 , a half of the user bandwidth (equal to b) is being wasted. In order to see this more clearly, at the bottom of Fig. 1, we have packed all the segments into two imaginary multiplexed channels, each with a normalized bandwidth of 1. If we also normalize the display durations as in (3.1), the shaded region at the bottom of Fig. 1 has area 1. Let A denote the area of the blank (wasted) region in these imaginary channels. Then we have $U(1+W) = 1 + A$, hence

$$W = (1+A)/U - 1, \quad (3.3)$$

where U is the normalized total user bandwidth. Eq. (3.3) implies that reducing A will reduce waiting time W . From our observation on wasted area, we have the following relation (recall $d_i = d_{i-1} + d_{i-2}$, hence $L_i = L_{i-1} + L_{i-2}$):

$$\begin{aligned} A &= 2*L_6 + L_5 \\ &= L_6 + 2*L_5 + L_4 \\ &= L_6 + L_5 + 2*L_4 + L_3 \\ &= L_6 + L_5 + L_4 + 2*L_3 + L_2 \\ &= L_6 + L_5 + L_4 + L_3 + 2*L_2 + L_1 \\ &= L_6 + L_5 + L_4 + L_3 + L_2 + 3*L_1 \end{aligned}$$

Substituting Eq. (3.2) in this equation, we arrive at

$$A = 1 + 2*L_1.$$

Substituting this and $U=2$ in Eq. (3.3), we finally obtain

$$W = L_1.$$

This is not surprising, if one looks at the download plane of Fig. 1 carefully. Now that we have a very simple formula for W , let us compute it for Fibonacci Broadcasting with different values of $N=n$. For a given value of $N=n$ in column 1, column 2 of Table 1 shows L_n as a multiple of L_1 . Note that $D*L_i = d_i$ (3.2).

Server Bandwidth Requirement ($N=n$)	Segment Size (as a multiple of L_1)	Waiting Time ($W=L_1$)
2	$L_2 : 2 L_1$	0.3333
3	$L_3 : 3 L_1$	0.1667
4	$L_4 : 5 L_1$	0.09091
5	$L_5 : 8 L_1$	0.05263
6	$L_6 : 13 L_1$	0.03125
7	$L_7 : 21 L_1$	0.01887
8	$L_8 : 34 L_1$	0.01149
9	$L_9 : 55 L_1$	0.007042
10	$L_{10} : 89 L_1$	0.004329
11	$L_{11} : 144 L_1$	0.002667
12	$L_{12} : 233 L_1$	0.001645

For each value of $N=n$ in column 1 in Table 1, column 3 shows the normalized waiting time. The entry at row 2 ($N=n=3$) and column 3, for example, was computed as follows. Since $L_3 = 3*L_1$ and $L_2 = 2*L_1$, we have $1 = L_3 + L_2 + L_1 = 6*L_1$, hence $L_1 = 1/6$ and $W = L_1 \approx 0.1667$. Similarly, the entry at row 6, column 3 was computed as follows: $1 = L_6 + L_5 + \dots + L_1 = 13*L_1 + 8*L_1 + \dots + L_1 = 32*L_1$, from which one obtains $W = L_1 = 1/32 \approx 0.03125$. The column 3 of Table 1 shows that increasing $N=n$ is very beneficial for reducing the waiting time.

Unfortunately, increasing $N=n$ 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 $N=n$, 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 N (number of channels) = 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 $1 \leq K \leq N$. After a fixed delay, $w (=D*W)$, the user can begin to display S_1 .

Table 1: Fibonacci Broadcasting.

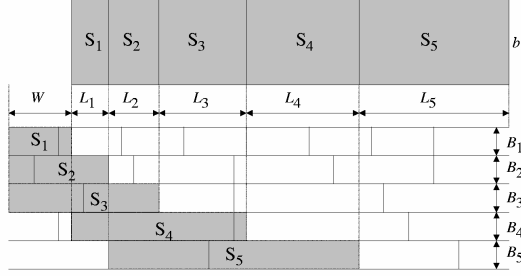


Figure 3. Illustration of Generalized Fibonacci Broadcasting (GFB).

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, ..., K . In general, for $i = 1, 2, \dots, n-1$, the user ceases to receive segment S_{i+1} when the previous data segment S_i has been completely displayed. By definition, we have $B_i = 1/g$ for all i , and the server bandwidth is simply

$$B_{GFB} = N/g \quad (3.4)$$

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

$$B_{GFB}^{User} = K/g \quad (3.5)$$

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

$$L_1 = (1/g) * W \quad (3.6)$$

$$L_2 = (1/g) * (W + L_1) = [1 + (1/g)] * L_1$$

$$L_3 = (1/g) * (W + L_1 + L_2) = [1 + (1/g)]^2 * L_1$$

.....

$$L_K = (1/g) * (W + L_1 + \dots + L_{K-1}) = [1 + (1/g)]^{K-1} * L_1 \quad (3.7)$$

$$L_{K+1} = (1/g) * (L_1 + \dots + L_K) = ([1 + (1/g)]^K - 1) * L_1 \quad (3.8)$$

$$L_{K+2} = (1/g) * (L_2 + L_3 + \dots + L_{K+1})$$

$$L_{K+3} = (1/g) * (L_3 + L_4 + \dots + L_{K+2})$$

.....

$$L_N = (1/g) * (L_{N-K} + L_{N-K+1} + \dots + L_{N-1})$$

The above recurrence relation resembles the generalized Fibonacci sequence. Hence the name Generalized Fibonacci Broadcasting. We use **GFB**(K/g) to denote GFB with parameters K and g . The total number of channels, $N=n$, 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 bandwidth equal to $1/g = 1$, and the user downloads data from $K=2$ concurrent channels, limiting the user bandwidth to $U=K/g=2$. Different choices for N lead to different waiting time W . Independent of the user request time, the waiting time for GFB is (from Eq. (3.6)):

$$W = g * L_1 \quad (3.9)$$

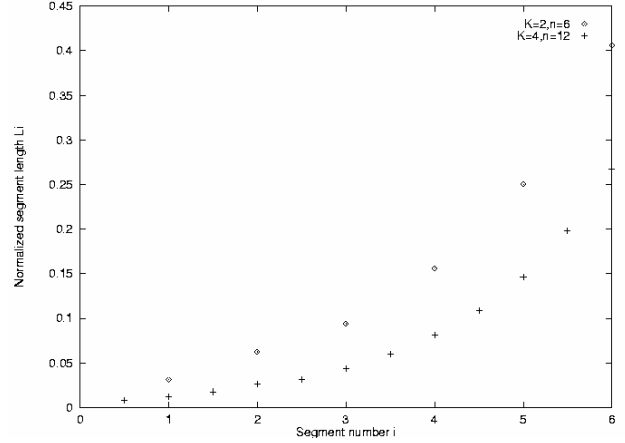


Figure 4: Normalized segment durations $\{L_i\}$ for the cases ($K=2, g=1, N=6$) and ($K=4, g=2, N=12$).

Fibonacci Broadcasting is clearly a special case of GFB, where $g=1$. In Fig. 4, $\{L_i\}$ for two different combinations of parameter values are plotted. The dots represent L_1, L_2, \dots, L_6 , when $K=2, g=1$, and $N=6$, computed from the above recurrence relation (they are also given in Table 1). Similarly, the +’s at 0.5, 1, 1.5, ... 5.5, and 6 represent L_1, L_2, \dots, L_{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 L_1 for the latter case is much less 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 , since g doubled but L_1 was reduced to much less than half (see Eq. (3.9)).

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 bandwidths and/or waiting times (W) of different schemes.

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

Fig. 5 compares FB-3 [16] and NPB-3 [16] (Sec. 2.1) with 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.

It is observed in [20] that, while NPB-3 [15] uses 249 segments to achieve a waiting time of $W \approx 0.004$ (equivalent to $w=28.9\text{sec}$ for a 120min video), GFB(6/2) with user bandwidth $U=6/2=3$ (shown in Fig. 9) needs only 15 segments (hence 15 channels, and the server bandwidth of $N/g = 15/2 = 7.5$) to achieve a shorter waiting time $W \approx 0.003$ (equivalent to $w=22.3\text{sec}$ for a 120min video). Although NPB-3 uses only 7 “channels,” the overheads associated with managing the channels are rather different between these two schemes. GFB simply repeats the same segment on each channel over and over again. Each “channel” of NPB-3, however, must multiplex a large number of segments *at different frequencies*. We elaborate on this when we compare FDPB-2 with GFB below. Note that the segments in NPB-3 are of a fixed size, while GFB’s segments are not. Fixed-size segments are an advantage if segments are multiplexed into a channel. However, since GFB doesn’t use multiplexing, its use of non-uniform segment lengths wouldn’t be a disadvantage. Similarly, we can show that GFB(10/(10/4)) outperforms both FB-4, NPB-4.

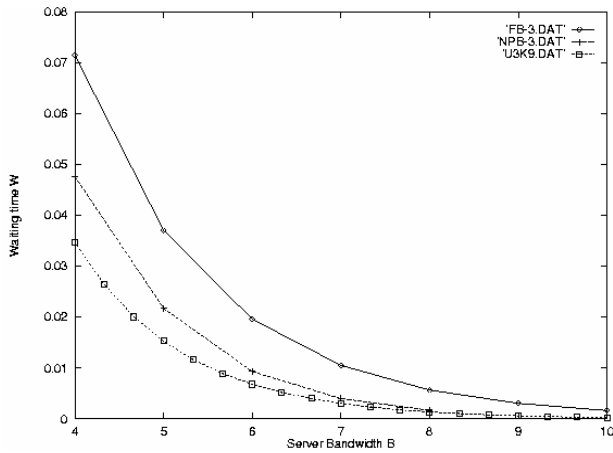


Figure 5. Waiting time W vs. server bandwidth B for FB-3, NPB-3, and GFB(9/3), from top to bottom.

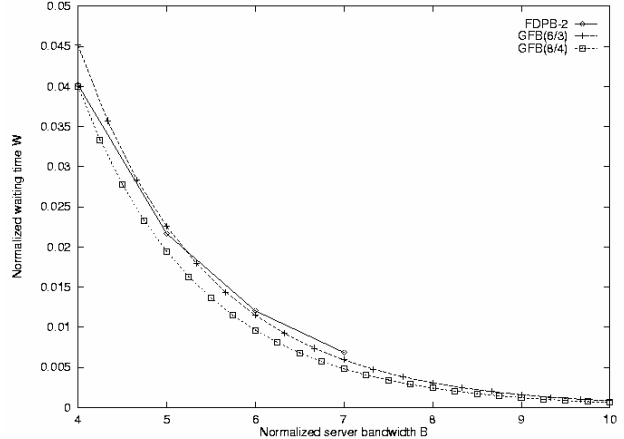


Figure 6. Waiting time W vs. server bandwidth B for FDPB-2 ($m=100$), GFB(6/3), and GFB(8/4), from top to bottom.

Fig. 6 compares, FDPB-2 (with $m = 100$) [Pâr01], GFB(6/3) and GFB(8/4), all of which limit the user bandwidth to 2. (The waiting times achieved by FDPB-2 for the server bandwidth above 7 were not available to us.) For example, for a server broadcasting bandwidth of 7, GFB(6/3) and FDPB-2 achieve a waiting time of 42.7sec and 49.3sec, respectively, for a 120min video. GFB(6/3) achieves this using $N = 7 * g = 21$ segments (i.e., 21 channels). Each “channel” of FDPB-2, however, must multiplex a large number (a total of 14,595 for the 7 “channels”) of segments *at different frequencies*, using 458 “subchannels. This may increase the implementation overhead considerably, especially if the segment size is small. Thus, we believe that managing the 21 channels of GFB(6/3) would require much less overhead. It is also observed that, with server broadcasting bandwidths of 4 and 5, FDPB-2 achieves shorter waiting times than GFB(6/3). A nice feature of GFB(K/g) 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 times than both FDPB-2 and GFB(6/3), but requires more user channels. FDPB-2 also has a parameter m , which can be increased to improve its performance. However, it will increase the number of segments even further.

4.2. Choosing parameter values 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.

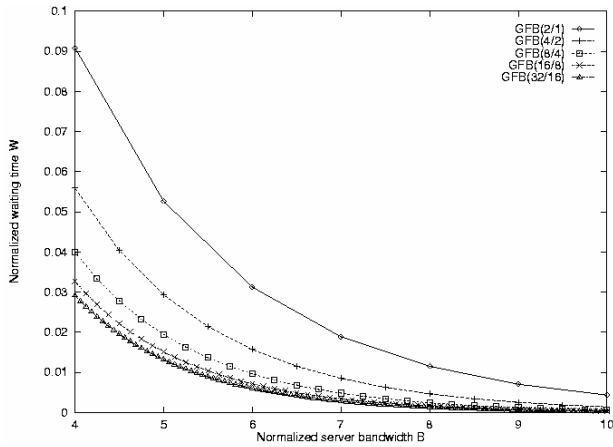


Figure 7. Waiting time W vs. server bandwidth B for GFB(K/g) with $K/g = 2$: (GFB(2/1), GFB(4/2), GFB(8/4), GFB(16/8), and GFB(32/16), from top to bottom.)

Fig. 8 is a similar graph with $K/g = 4$. From Figs. 7 and 8, we can see that if we keep the user bandwidth $U = K/g$ constant, increasing g and K proportionally leads to reduced server bandwidth requirement 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 $U \geq 4$; larger for $U=2$). For the bottom two overlapping curves, $g \geq 10$ 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.

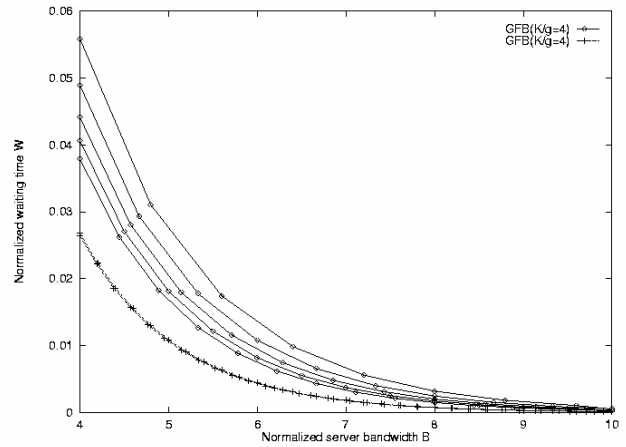


Figure 8. Waiting time W vs. server bandwidth B for GFB(K/g) with $K/g = 4$: (GFB(5/(5/4)), GFB(6/(3/2)), GFB(7/(7/4)), GFB(8/2), GFB(9/(9/4)), GFB(20/5), and GFB(21/(21/4)), from top to bottom.)

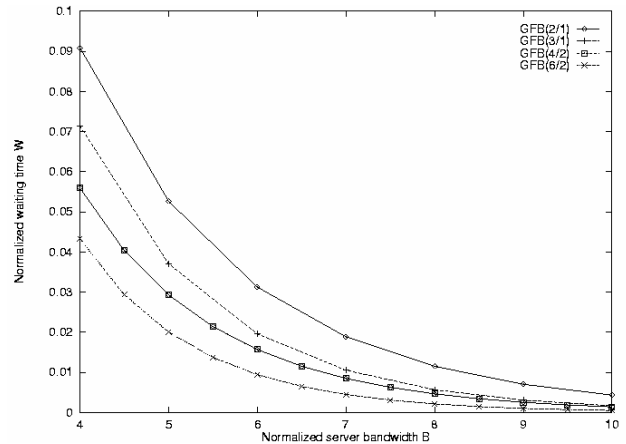


Figure 9. Comparison of GFB(2/1), GFB(3/1), GFB(4/2), and GFB(6/2).

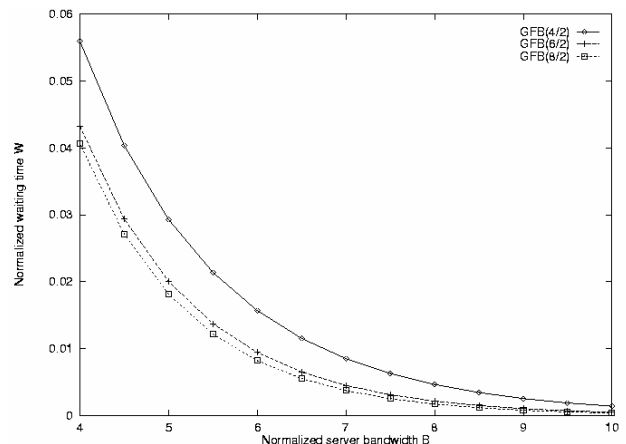


Figure 10. Waiting time W vs. server bandwidth for GFB($K/2$) (GFB(4/2), GFB(6/2) and GFB(8/2), from top to bottom.)

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 the user downloads from all available broadcast channels concurrently ($K=N$), GFB coincides with Greedy Equal Bandwidth Broadcasting [4] (see Sec. 2.2).

4.3. Asymptotic lower bounds

One would naturally be interested in the limit of performance of GFB(K/g) as K and g approached infinity, while keeping $U = K/g$ fixed, e.g., the lower bound for all the curves in Figs. 7 and 8.

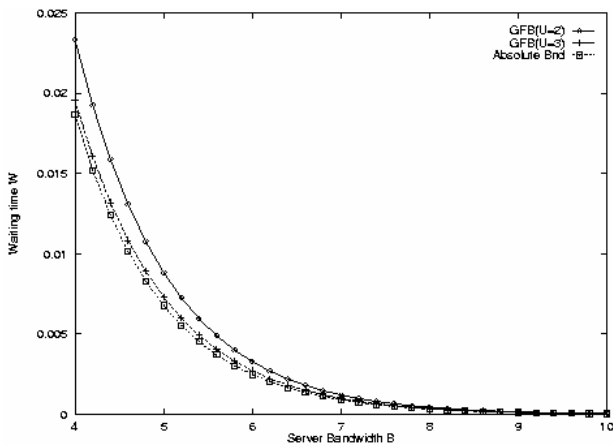


Figure 11. Asymptotic lower bounds.

According to [3] and [Pär01], the absolute lower bound on waiting time W for all schemes is given by

$$W = 1/(e^B - 1) \quad (4.1)$$

This curve is plotted in Fig. 11 (see the bottom curve with square data points). With some tedious math, we can compute the limiting case for GFB from the recurrence relation given in Sec. 3. We find that

$$W \rightarrow 1/[e^B - 1 - 2(e^{B-U} - 1) + (B-U)] \quad \text{for } U \leq B \leq 2U, \quad (4.2)$$

and

$$W \rightarrow 1/[e^B - 1 - 2(e^{B-U} - 1) + (e^{B-2U} - 1) + U] \quad \text{for } B > 2U. \quad (4.3)$$

Using these formulae, we have plotted the cases $U=2$ and $U=3$ in Fig. 11. It is seen that, for the case $U=3$ (the middle curve), in particular, the limiting case of GFB is rather close to the absolute lower bound.

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 $1 \leq K \leq N$. In GFB(K/g), the server uses N channels, broadcasting each video segment on a dedicated channel with bandwidth b/g , where b (Mbits/sec) is the display rate. The user downloads data from at most K out of the N channels. Therefore, the user bandwidth is limited to $K*b/g$, and the server bandwidth is $N*b/g$. This gives a service provider great flexibility in choosing the appropriate parameter values, according to the technology (and the funds) 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 $K*b/g$ 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 sub-channels relatively easily.

Last but not least, GFB gives fine-grained choices for the server and user bandwidths, provided by different combinations of K and g .

It is clear that GFB can be combined easily with the idea of “aggressive preloading” [17] (prefix caching) to reduce the waiting time and/or the server bandwidth.

6. References

- [1] C.C. Aggarwal, J.L. Wolf, and P.S. Yu, “A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems”, *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems*, Hiroshima, Japan, June 1996, pp. 118-126.
- [2] C.C. Aggarwal, J.L. Wolf, and P.S. Yu, “Design and Analysis of Permutation-Based Pyramid Broadcasting”, *Multimedia Systems* 7(6), 1999, pp. 439-448.
- [3] A. Hu, “Video-on-Demand Broadcasting Protocols: A Comprehensive Study”, *Proc. INFOCOM '01*, Anchorage, AK, April 2001.

- [4] A. Hu, I. Nikolaidis, and P. Beek, "On the Design of Efficient Video-on-Demand Broadcast Schedules", *Proc. 7th Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MAS-COTS'99)*, 1999, pp. 262-269.
- [5] K.A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems", *ACM SIGCOMM'97*, Cannes, France, September 1997, pp. 89-100.
- [6] L. Juhn and L. Tseng, "Harmonic Broadcasting for Video-On-Demand Service", *IEEE Trans. on Broadcasting* 43(3), Sept. 1997, pp. 268-271.
- [7] L. Juhn and L. Tseng, "Fast Broadcasting for Hot Video Access", *Proc. 4th Int'l Workshop on Real-Time Computing Systems and Applications*, Oct. 1997, pp. 237-243.
- [8] L. Juhn and L. Tseng, "Staircase Data Broadcasting and Receiving Scheme for Hot Video Service", *IEEE Trans. on Consumer Electronics* 43(4), Nov.1997, pp. 1110-1117.
- [9] L. Juhn and L. Tseng, "Fast Data Broadcasting and Receiving Scheme for Popular Video Service", *IEEE Trans. on Broadcasting* 44(1), Mar. 1998, pp. 100-105.
- [10] L. Juhn and L. Tseng, "Enhanced Harmonic Data Broadcasting and Receiving Scheme for Popular Video Service", *IEEE Trans. on Consumer Electronics* 44, May 1998, pp. 343-346.
- [11] J.-F. Pâris, "A Fixed-Delay Broadcasting Protocol for Video-On-Demand", *Proc. 10th Int'l Conf. on Computer Communications and Networks 2001*, 2001, pp. 418-423.
- [12] J.-F. Pâris, S.W. Carter, and D.D.E. Long, "A Low Bandwidth Broadcasting Protocol for Video on Demand", *Proc. IEEE Int'l Conf. on Computer Communications and Networks (IC3N'98)*, 1998, pp. 690-697.
- [13] J.-F. Pâris, S.W. Carter, and D.D.E. Long, "Efficient Broadcasting Protocols for Video on Demand", *Proc. 6th Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98)*, July 1998, pp. 127-132.
- [14] J.-F. Pâris, S.W. Carter, and D.D.E. Long, "A Hybrid Broadcasting Protocol for Video on Demand", *Proc. 1999 Multimedia Computing and Networking Conference (MMCN'99)*, San Jose, CA, Jan. 1999, pp. 317-326.
- [15] J.-F. Pâris, "A Simple Low-Bandwidth Broadcasting Protocol for Video-on-Demand", *Proc. 8th Int'l Conf. on Computer Communications and Networks (IC3N'99)*, Boston-Natick, MA, October 1999, pp. 118-123.
- [16] J.-F. Pâris and D.D.E. Long, "Long, Limiting the User Bandwidth of Broadcasting Protocols for Video-on-Demand", *Proc. Euromedia 2000 Conf.*, Antwerp, Belgium, May 2000, pp. 107-111.
- [17] J.-F. Pâris and D.D.E. Long, "The Case for Aggressive Partial Preloading in Broadcasting Protocols for Video-on-Demand", *Proc. 2001 IEEE Int'l Conf. On Multimedia and (ICME2001)*, Tokyo, Aug. 2001, pp. 113-116.
- [18] S. Viswanathan and T. Imielinski, "Pyramid Broadcasting for Video on Demand Service", *Proc. IEEE Multimedia Computing and Networking Conf.*, Vol. 2417, San Jose, California, 1995, pp. 66-77.
- [19] S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting", *Multimedia Systems* 4(4): 197-208, Aug. 1996.
- [20] E.M. Yan, "Minimizing Bandwidth Requirement of Broadcasting Protocol in Video-on-Demand Services", *M.Sc. Thesis*, School of Computing Science, Simon Fraser University, April 2002.