A Borrow-and-Return Model to Reduce Client Waiting Time for Broadcasting-Based VOD Services

Ming-Hour Yang, Chi-He Chang, and Yu-Chee Tseng

***Abstract*—One way to broadcast a popular video is to use multiple channels, each broadcasting a portion of the video periodically. Among the many schemes falling in this category, this paper focuses on several representative schemes (such as FB [15], [18], Pagoda [22], [24], and RFS [26]), which all share a *FSFC* property by repeatedly broadcasting the First Segment of the video on the First Channel. We propose a general *Borrow-and-Return model* that can be immediately applied to any scheme owning the FSFC property to reduce the viewer's waiting time without increasing the number of channels required. Given a group of videos, the basic idea is to lend the free time slots of videos without viewers to those videos with viewers to speedup the latter's transmission. By so doing, some bandwidth may be vacated by the borrowing videos to benefit others' transmission. Effectiveness of this model is analyzed by applying it to the FB scheme.**

***Index Terms*—Broadcasting, cable TV, channel allocation, communication, digital video broadcasting, video-on-demand (VOD).**

I. Introduction

VIDEO has become one of the most important media in our lives. On average, each household has 1.4 television sets in the U.S. [3]. With the availability of networks, people may wish to access videos at the touch of their fingertips instantly. This has motivated the efforts of providing VOD (video-on-demand) services [19], [21]. Offering such services is likely to be popular at local residential areas, and viable in metropolitan areas in the near future.

A VOD system is typically implemented by a client-server architecture supported by certain transport networks such as telecom, CATV, or satellite networks [5], [12], [25]. The simplest scheme is to dedicate a channel to each client [9], [20]. Many VCR-like functions may be simulated (e.g., forward, rewind, pause, search, etc.). Since video is an isochronous medium, the video server has to reserve a sufficient amount of network bandwidth and I/O bandwidth for each video stream before committing to a client's request [10]. However, such systems may easily run out of channels because the growth of network bandwidths may never keep up with the growth of clients.

To relieve the stress on the bandwidth and I/O demands, many alternatives have been proposed by sacrificing some VCR functions. The *batching* approach collects a group of requests that arrive close in time, and serves them all together with one channel [1], [7], [8]. A stream tapping scheme [4] is proposed to allow a client to greedily tap data from any stream on the VOD server containing data the client can use. A scheduling policy based on the arrival of requests is required to best utilize the channels. Two *patching* schemes [11], [13] that are similar to stream tapping are proposed on top of the batching approach to allow late-coming clients to join the service under some buffer and channel constraints. If the video is very popular/hot, the *broadcasting* approach will be more efficient [2], [6]–[8], [10], [14]–[18], [22], [24], [27]. In such schemes, the server uses multiple dedicated channels cooperatively to broadcast a video. Each channel is responsible of broadcasting some portion of the video periodically. Each client follows some reception rule to retrieve data from appropriate channels so as to play the whole video continuously. Popular/hot videos are likely to interest many viewers at a certain period of time. According to [7], [8], 80% of demands are on a few (10 or 20) very popular videos.

Broadcasting schemes are reviewed below. The simplest solution is to periodically broadcast the video on several channels, each differentiated by some time [6]. The EB scheme [7] proposes to divide the video into equal-length segments; a user has to wait no longer than the length of one segment. Many schemes have been proposed by imposing a larger client receiving bandwidth and an extra buffering space at the client side. The *pyramid* scheme [27] can reduce the maximum waiting time experienced by viewers exponentially with respect to the number of channels used. The pyramid scheme is further improved by the *permutation-based pyramid* scheme [2], *skyscraper* scheme [14], and *greedy disk-conserving* scheme [10] to address the disk buffering requirement at the client side. A number of works have dedicated to reducing the waiting time experienced by viewers. Two instances are the Fast Broadcasting (FB) scheme [15], [18] and the Pagoda scheme [22], [24], which can broadcast a video using k channels by having new-coming viewers to wait no longer than $\Theta(D/2^k)$ and $\Theta(D/5^{k/2})$ time, respectively, where D is the length of the video. A *harmonic* scheme based on the concept of harmonic series is proposed in [16], [17]. A scheme called *quasiharmonic* is proposed by reorganizing server's broadcasting sequence in a stream [23].

Among the many broadcasting schemes, we focus on the schemes in [15], [18], [22], [24], [26] which are characterized by low viewer's waiting time and yet easy to implement. We observe that they all share the same *FSFC* property that the First Segment of the video is repeatedly broadcast on the First Channel. This is done no matter there are viewers coming

Manuscript received October 16, 2001; revised February 21, 2003. This work is co-sponsored by the Lee and MTI Center for Networking Research at the National Chiao Tung University and the MOE Program for Promoting Academic Excellence of Universities under Grants A-91-H-FA07-1-4 and 89-E-FA04-1-4.

The authors are with the Department of Computer Science and Information Engineering, National Chiao Tung University, Hsin-Chu, 30050, Taiwan, R.O.C. (e-mail: yctseng@csie.nctu.edu.tw).

Digital Object Identifier 10.1109/TBC.2003.813439

for the video or not. We thus propose a *Borrow-and-Return model*, which can further improve viewer's waiting time. The basic idea is to lend the free time slots of those videos without viewers to those videos with new-coming viewers. This will speedup the latter's broadcasting speed and vacate some bandwidth. Later on, if there are new-coming viewers, the vacated bandwidth may be used to serve these clients at earlier time. Two types of borrowing, namely *SBML (single borrower, multiple lenders)* and *MBML (multiple borrower, multiple lenders)*, are proposed. Analyses are presented to show the effectiveness of the proposed model. It is worth noting that this model can be applied to any broadcasting scheme that has the FSFC property.

The rest of this paper is organized as follows. In Section II, we review two broadcasting schemes, FB and Pagoda, that own the FSFC property. Our Borrow-and-Return model is presented in Section III. Analyses are in Section IV. Conclusions are drawn in Section V.

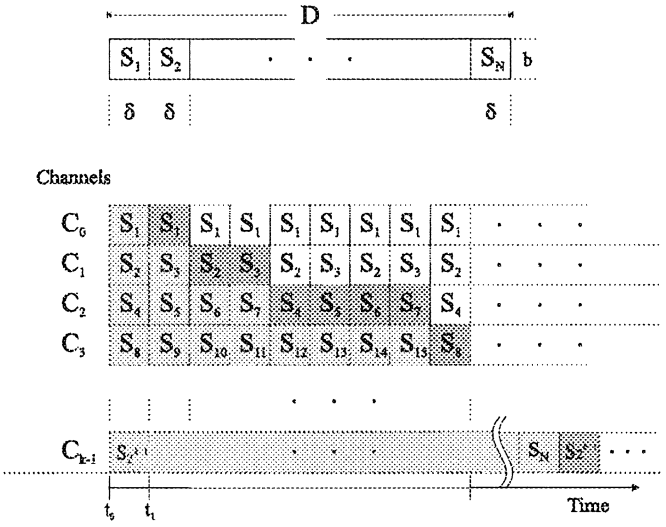


Fig. 1. The Fast Broadcasting (FB) scheme.

II. Reviews

To give some feeling of the broadcasting approach, below we review two representative broadcasting schemes.

A. The Fast Broadcasting (FB) Scheme

In the Fast Broadcasting (FB) scheme [15], [18], we are given a video V of length D which requires a transmission bandwidth of b (for instance, V could be a high-quality MPEG-II-compressed NTSC video of length $D = 120$ minutes to be played at rate $b \approx 10$ Mbps). Since it is assumed that V is a popular video, providing each client a dedicated channel to view V is infeasible. To relieve the demand on channels, we can use k channels, $C_0, C_1, \ldots, C_{k-1}$, each of bandwidth b, to broadcast V. Contents of V will be arranged on these channels according to the following rules.

1) Partition V evenly into n segments, $S_1, S_2, \ldots, S_n$, where $n = 2^k - 1$. That is, the concatenation $S_1 \circ S_2 \circ \cdots \circ S_n = V$ (we denote by $\circ$ the concatenation operator). The length of each segment is thus $\delta = D/n = D/(2^k - 1)$.
2) Divide each channel $C_i, i = 0, \ldots, k-1$, into time slots of length δ. On C_i, broadcast data segments $S_{2^i}, S_{2^i+1}, \ldots, S_{2^{i+1}-1}$ periodically and in that order. Note that the first segment S_{2^i} of each $C_i, i = 0, \ldots, k-1$, should be aligned in the same time slot.

An example is in Fig. 1. Channel C_0 broadcasts the first segment periodically, C_1 broadcasts the next two segments periodically, C_2 broadcasts the next four segments periodically, etc.

To view V, a client should receive video contents from all k channels concurrently according to the following rules.

1) To start the service, wait until the beginning of *any* new time slot.
2) Concurrently from each channel $C_i, i = 0, \ldots, k-1$, download 2^i consecutive segments starting from the first time slot.
3) Right at the moment when step 2 begins, start to consume the video $S_1 \circ S_2 \circ \cdots \circ S_n$.

For example, in Fig. 1 suppose we allocate $k = 4$ channels for V. So V is partitioned into $n = 2^4 - 1 = 15$ segments. For a client starting at time t_0, it will receive segments S_1, S_2, S_4, S_8 from C_0, C_1, C_2, C_3, respectively, in the first time slot. During the first time slot, segment S_1 will be consumed, and the other premature segments S_2, S_4, S_8 will be buffered at the client's local storage for future use. In the second slot, the client will consume segment S_2 from its local storage. At the same time, segments S_3, S_5, S_9 from C_1, C_2, C_3, respectively, will be buffered. In the third time slot, the client will consume the S_3 from its local storage, and simultaneously buffer S_6 and S_{10} from C_2 and C_3, respectively. This will be repeated until the client has received $2^3 = 8$ data segments from C_3. At last, the client will finish watching the video at time $t_0 + n\delta = t_0 + 15\delta$. It is not hard to derive similar conclusion if the client starts at other time slots.

In some special time slots, it is possible for a client to play the video without buffering. For instance, if a client starts at time t_1 of Fig. 1, it can continuously receive every required segment (the darker segments in the figure) just-in-time from one of the channels. However, this happens only once every 2^{k-1} time slots.

In summary, the FB scheme allows a client to start at the beginning of any time slot by ensuring that whenever a segment is needed to be consumed, either it has been buffered previously or it is being broadcast *just-in-time* on one of the channels. We briefly outline the proof as follows. Suppose that a client begins to download S_1 at time t. Consider the 2^i segments $S_{2^i}, S_{2^i+1}, \ldots, S_{2^{i+1}-1}$, which are periodically broadcast on $C_i, i = 0, \ldots, k-1$. These segments will be downloaded by the client from C_i in the time interval $[t, t + 2^i\delta]$. However, these segments will be viewed by the client in the time interval $[t + (2^i - 1)\delta, t + (2^{i+1} - 1)\delta]$. There is only one slot of overlapping, i.e., $[t+(2^i-1)\delta, t+2^i\delta]$, between the above two time intervals. In this time slot, S_{2^i} is the segment to be played. It can be easily observed that S_{2^i} either has appeared on C_i previously, or is currently being broadcast on C_i in time. This concludes the proof.

What the FB scheme achieves is to shorten viewer's maximum waiting time with only a few channels. A client has to wait

no longer than δ time to start viewing the video. The average waiting time is $\delta/2$. Since $\delta = D/(2^k - 1)$, a small increase in k can reduce the waiting time significantly. For instance, given a 120-minute video, with 5 channels, a viewer has to wait no more than $120/(2^5 - 1) < 4$ minutes to start the service, and with 6 channels, the maximal waiting time further reduces to $120/(2^6 - 1) < 2$ minutes.

B. The Pagoda Scheme

The Pagoda scheme [22], [24] can further reduce viewer's waiting time. Still, we are given a video V of length D and k channels, $C_0, C_1, \ldots, C_{k-1}$. The video server uses the following rules to broadcast V:

1) Partition V evenly into n segments, $S_1, S_2, \ldots, S_n$, where $n = 4(5^{(k/2)-1}) - 1$ if k is even, and $n = 2(5^{\lfloor k/2 \rfloor}) - 1$ if k is odd. Also, divide each channel into time slots of fixed length $\delta = D/n$.
2) On C_0, broadcast segment S_1 periodically. For $j = 1, \ldots, \lfloor (k-1)/2 \rfloor$, periodically broadcast on channel C_{2j-1} the segments

$$S_z, S_{2z}, S_{z+1}, S_{2z+2}, \ldots, S_{(3z/2)-1}, S_{3z-2}, S_z, S_{2z+1}, S_{z+1}, S_{2z+3}, \ldots, S_{(3z/2)-1}, S_{3z-1},$$

and on channel C_{2j} the segments

$$S_{3z/2}, S_{3z}, S_{4z}, \ldots, S_{2z-1}, S_{4z-2}, S_{5z-2}, S_{3z/2}, S_{3z+1}, S_{4z+1}, \ldots, S_{2z-1}, S_{4z-1}, S_{5z-1},$$

where $z = 2(5^{j-1})$.

3) If k is even, then periodically broadcast on the last channel C_{k-1} the segments

$$S_z, S_{z+1}, \ldots S_{2z+1},$$

where $z = 2(5^{j-1})$.

Fig. 2 shows the Pagoda scheme's scheduling for $k = 3$ and 4. The video will be partitioned into $2(5^{\lfloor 3/2 \rfloor}) - 1 = 9$ and $4(5^{(4/2)-1}) - 1 = 19$ segments, respectively. In the figure, we mark by gray when and where to grab the necessary segments for a client starting at the first time slot. The reader may refer to [22], [24] for more details of the Pagoda scheme, where it has been proved that as long as segment S_i is broadcast by a period no less than once per i time slots, no disruption will be experienced by any viewer.

To summarize, the number of segments (n) in Pagoda will grow much faster as k increases than that of FB. So the waiting time (i.e., D/n) can be significantly reduced. For instance, with 8 channels, a video will be partitioned into 499 segments by Pagoda, and 254 segments by FB.

III. The Borrow-and-Return Model

In the following, we first introduce the basic idea. Then we present our two schemes, called SBML and MBML.

A. Basic Idea

We observe that many broadcasting schemes, including the above reviewed two, own the FSFC property. The first segment is repeatedly broadcast in the first channel. All new-coming viewers have to wait for the beginning of a time slot to start playing the video. However, in the beginning of a time slot, if the video server finds that there are no new-coming viewers waiting to receive this video, broadcasting the first segment in the coming time slot will be wasteful. Thus, rather than broadcasting a useless segment, this time slot can be "lent" to other videos to speedup their broadcasting. By so doing, some bandwidths will be released at earlier time. This may be used to benefit new-coming viewers in the future, which is regarded as a "return."

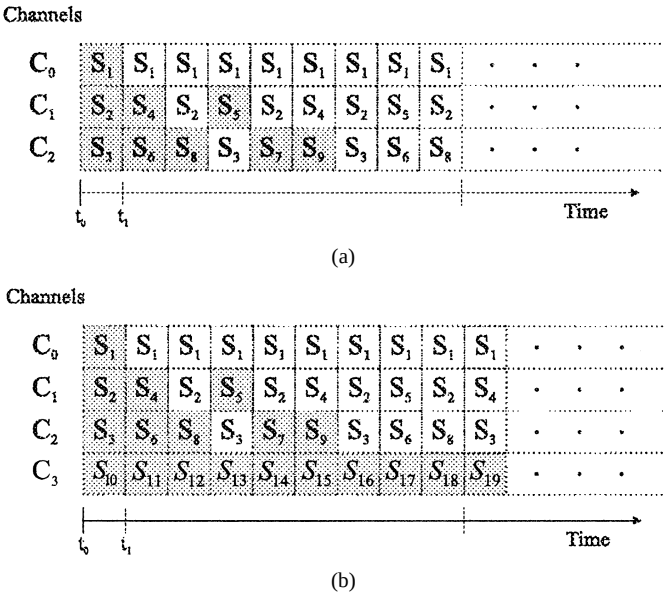


Fig. 2. The Pagoda scheme: (a) $k = 3$ channels, and (b) $k = 4$ channels.

For example, consider two videos V and V'. Suppose C and C' are their first channels, respectively, as shown in Fig. 3. Let the first segments of V and V' be denoted as S and S', respectively. Suppose that there are no viewers arriving in the time slot $[t_0, t_1]$ for V. Then broadcasting S in time slot $[t_1, t_3]$ will be useless. We can lend this time slot to V'. This will speedup the broadcasting of S', making it complete at earlier time of t_2. Now, in time interval $[t_2, t_3]$, both channels C and C' are free. These bandwidths may be used to serve new-coming viewers arriving during the interval $[t_1, t_2]$.

We comment that although the broadcast-based schemes are targeted at very popular videos, the level of hotness of a video is still likely to change by time of a day. Our Borrow-and-Return model is more appropriate for systems with very dynamic loads. In fact, our result can be regarded as a compliment to the broadcast-based schemes when the arrival of customers is so dynamic that it is sometimes even below the expected rate.

Below, we consider the borrow-and-return behavior among a group of videos. We propose two schemes.

B. Single Borrower, Multiple Lenders (SBML)

The scenario can be described as follows. Suppose there are m movies $V_1, V_2, \ldots, V_m$, each of the same length D and each being assigned k channels to broadcast. Let the jth channel assigned to V_i be C^i_{j-1} and the jth segment of V_i be S^i_j. Let δ be

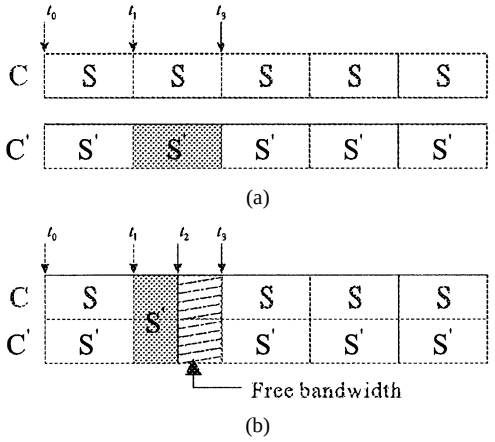


Fig. 3. Basic idea of the borrow-and-return model: (a) before borrowing, and (b) after borrowing.

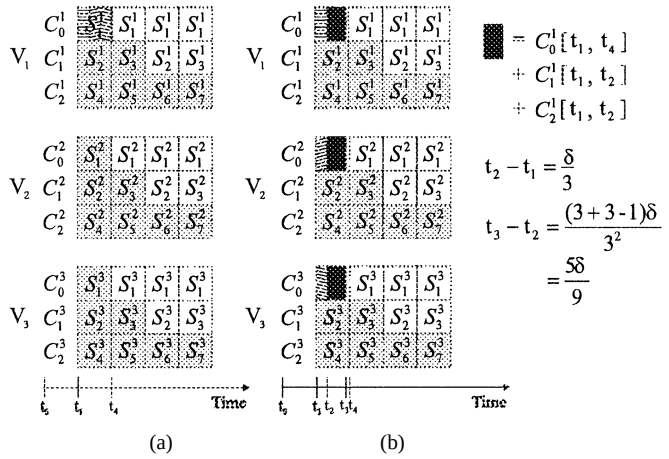


Fig. 4. SBML behavior based on the FB scheme: (a) before borrowing, and (b) after borrowing.

the length of a time slot (the value of δ differs for different broadcasting schemes). Fig. 4(a) illustrates an example with $m = 3$ and $k = 3$ based on the FB scheme.

Let t_1 be the beginning of a time slot. Suppose that there are viewers arriving in the previous time slot $[t_0, t_1]$ for V_1, but no viewers for $V_2, V_3 \ldots, V_m$. Then we can use the first channels of all m videos (i.e., $C_0^1, C_0^2, \ldots, C_0^m$) to cooperatively broadcast the first segment S_1^1 of V_1. The broadcasting will be completed at time $t_2 = t_1 + \delta/m$. [See Fig. 4(b) for an illustration.]

Now in the interval $[t_2, t_4]$, where t_4 is the ending time of the current time slot, all the first channels C_0^i, $i = 1 \cdots m$, will be free for use. Here, we can greedily pick one in $V_1, V_2, \ldots, V_m$ such that during the interval $[t_1, t_2]$, there are most new viewers arriving. Without loss of generality, let V_1 be the one with most new viewers. We will use these m channels cooperatively to serve these new viewers. (Note that under the normal situation, these viewers have to wait until t_4.) Since these viewers arrive later than time t_1 and do not receive until time t_2, the following video contents should be made up to them:

i) the normal contents of C_0^1 during interval $[t_1, t_4]$, and
ii) the normal contents of $C_1^1, C_2^1, \ldots, C_{k-1}^1$ during the interval $[t_1, t_2]$.

By "normal," we mean the video contents under the situation when no borrow-and-return activity happens. [For example, in the case of Fig. 4(a), the following contents should be made up to those new viewers: content of C_0^1 during $[t_1, t_4]$, content of C_1^1 during $[t_1, t_2]$, and content of C_2^1 during $[t_1, t_2]$.] Using m channels cooperatively, we can finish broadcasting these contents at time

$$t_3 = t_2 + \frac{\delta + \frac{\delta}{m} \times (k-1)}{m} = t_2 + \frac{(k+m-1) \times \delta}{m^2}.$$

However, note that we must guarantee the condition $t_3 \leq t_4$ hold, because otherwise this will interfere the broadcasting activity in the next time slot. That is, we will only serve the new-coming viewers of V_0 if $t_3 \leq t_4$ holds. For example, in Fig. 4(b), with $k = 3$ and $m = 3$, we have $t_3 = t_2 + (5/9)\delta = t_1 + (8/9)\delta$. This ensures that $t_3 \leq t_4$.

If the above succeeds, we will still have m free channels during period $[t_3, t_4]$. Again, we can greedily pick the video with the most new-coming viewers from $V_1, V_2, \ldots, V_m$ during the interval $[t_1, t_3]$ and serve them using the free bandwidth, if possible. This can be repeated recursively, until the current time slot expires.

C. Multiple Borrowers, Multiple Lenders (MBML)

Still, we assume m movies $V_1, V_2, \ldots, V_m$, each of the same length D and each being allocated k channels to broadcast. Let C_{j-1}^i, S_j^i, and δ the jth channel of V_i, the jth segment of V_i, and the length of a time slot, respectively.

Let t_1 be the beginning of a time slot. Suppose that there are viewers arriving in the previous time slot $[t_0, t_1]$ for videos $V_1, V_2, \ldots, V_r$, but no viewers for the other videos $V_{r+1}, V_{r+2}, \ldots, V_m$. Then we can use all channels C_0^i, $i = 1 \cdots m$, cooperatively to broadcast the segments S_1^i, $i = 1 \cdots r$. The broadcasting will be completed at time $t_2 = t_1 + (r\delta/m)$. (See Fig. 5 for an example with $r = 2$, $m = 4$, and $k = 3$ based on the FB scheme.)

Now in the interval $[t_2, t_4]$, where $t_4 = t_1 + \delta$ is the ending time of the current time slot, all channels C_0^i, $i = 1 \cdots m$, will be free for use. Again, we can greedily pick one in $V_1, V_2, \ldots, V_m$ such that during the interval $[t_1, t_2]$, there are most new viewers arriving. Without loss of generality, let V_1 be the one with most new viewers. Still, we will use m channels together to serve these new-coming viewers. Specifically, the following video contents should be made up to them:

i) the normal contents of C_0^1 during interval $[t_1, t_4]$, and
ii) the normal contents of $C_1^1, C_2^1, \ldots, C_{k-1}^1$ during the interval $[t_1, t_2]$.

These contents can be broadcast using m channels cooperatively by time

$$t_3 = t_2 + \frac{\delta + \frac{r\delta}{m} \times (k-1)}{m} = t_2 + \frac{(m + rk - r) \times \delta}{m^2}.$$

So, if $t_3 \leq t_4$, these new-coming viewers can be served successfully without interfering the activity in the next time slot. Otherwise, the cooperative broadcasting will be canceled. For example, in Fig. 5(b), with $k = 3$, $m = 4$, and $r = 2$, we have $t_3 = t_2 + (\delta/2) = t_1 + \delta$.

In the above example, $t_3 = t_4$. So no borrow-and-return can happen after t_3. In case that $t_3 < t_4$, we will still have

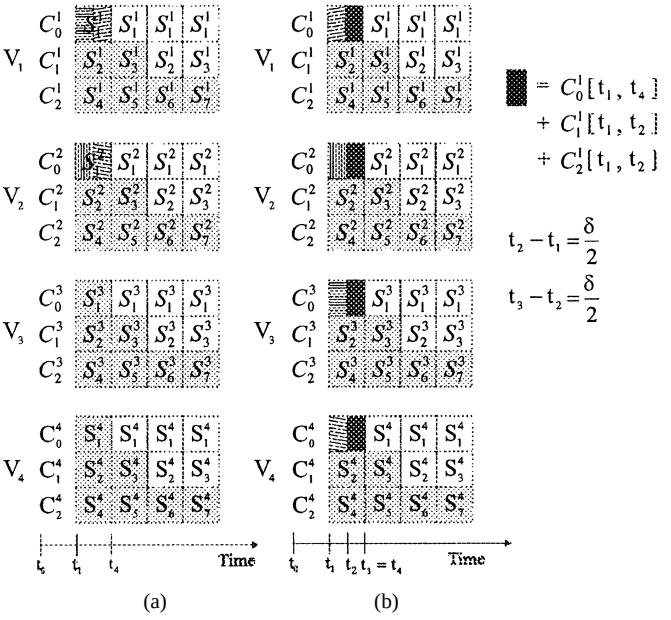


Fig. 5. MBML behavior ($r = 2$) based on the FB scheme: (a) before borrowing, and (b) after borrowing.

m free channels during period $[t_3, t_4]$. Similar to SBML, we can schedule the next video with the most new-coming viewers during $[t_1, t_3]$ to be served. This can be repeated recursively, until the current time slot expires.

To compare, with a single borrower, the SBML can complete the borrower's job at earlier time. But with multiple borrowers, since the available bandwidths are the same, the MBML will complete this with more time. This may result in less borrow-and-return activities for MBML, as shown in the above examples (Figs. 4 and 5). However, the advantage of MBML is that by starting at later time, it may potentially accumulate more viewers and serve them all at once. These tradeoffs will be analyzed and compared in the next section.

IV. Analysis and Comparison

This section analyzes the viewer's waiting time saved by SBML and MBML. Throughout the analysis, we assume that there are totally n videos $V_1, V_2, \ldots, V_n$, all of the same length D. Let the viewer arrival rate for V_i be λ_i based on the Poisson distribution, and assume without loss of generality that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Also, assume that for each video, k channels are assigned to it.

A. Waiting Time of SBML

Consider any time slot. Let B be the set of m videos such that there are no new-coming viewers for them during the previous time slot. The probability of this is

$$NN(B) = \left(\prod_{V_j \in B} e^{-\lambda_j \delta}\right) \times \left(\prod_{V_j \notin B} (1 - e^{-\lambda_j \delta})\right).$$

In SBML, we will use m empty slots of videos in B to serve a video not in B. The service time will be $W_0 = \delta/(m+1)$. According to SBML, we are likely to serve the new-coming viewers for V_1 first, followed by V_2, then V_3, ..., until the current time slot expires. Let's define the service time for V_j, $j \geq 1$, to be W_j,

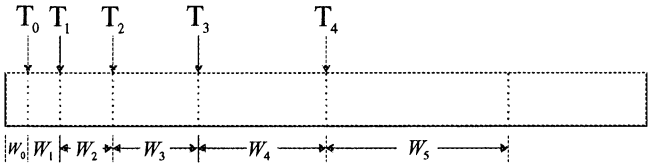


Fig. 6. The service time W_i.

$$W_j = \frac{\delta + (k-1)\sum_{c=0}^{j-1} W_c}{m+1}.$$

Intuitively, W_j is the time to make up the video contents for those new-coming viewers for V_j (which contains the complete segment of its first channel and the partial segments of the other channels from the beginning of the time slot until we start to serve it). These values are illustrated in Fig. 6.

Let's write the accumulated time $T_c = \sum_{j=0\cdots c} W_j$. It can be derived that

$$T_c = \frac{\delta}{m+1} + \frac{(k+m)\delta}{(m+1)^2} + \frac{(k+m)^2\delta}{(m+1)^3} + \cdots + \frac{(k+m)^c\delta}{(m+1)^{c+1}} = \frac{((k+m)^{c+1} - (m+1)^{c+1})\delta}{(k-1)(m+1)^{c+1}}.$$

Given a set B, the number of videos that can receive the borrow-and-return service in the current time slot, denoted by $p(B)$, must satisfy

$$T_{p(B)} \leq \delta < T_{p(B)+1}.$$

So we can determine the value of $p(B)$ by

$$\frac{\left((k+m)^{p(B)+1} - (m+1)^{p(B)+1}\right)\delta}{(k-1)(m+1)^{p(B)+1}} \leq \delta < \frac{\left((k+m)^{p(B)+2} - (m+1)^{p(B)+2}\right)\delta}{(k-1)(m+1)^{p(B)+2}}.$$

For those $p(B)$ videos that are served, the aggregated amount of saving in waiting time is

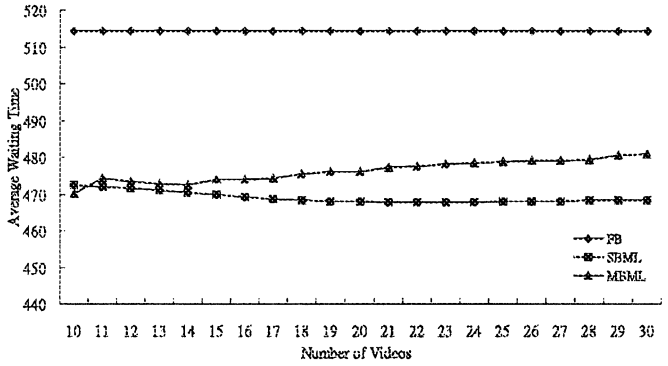
$$SA(B) = \sum_{j=1}^{p(B)} \lambda_j T_{j-1} \cdot (\delta - T_{j-1}). \quad (1)$$

Taking different combinations of B into consideration, we derive the average viewer waiting time as

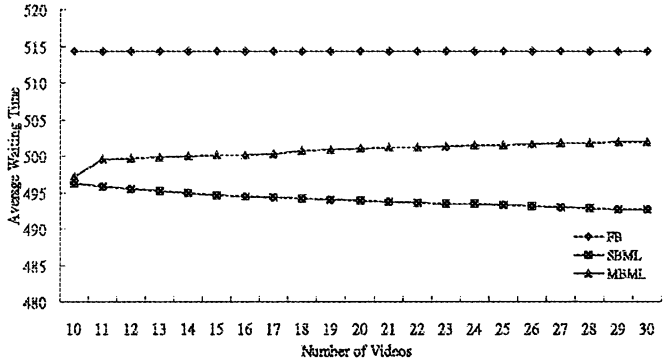
$$T_{SBML} = \frac{\sum_{j=1}^{n} \lambda_j \delta \cdot \delta/2 - \sum_{B \subseteq \{V_1, \ldots V_n\}} NN(B) \cdot SA(B)}{\sum_{j=1}^{n} \lambda_j \delta}.$$

B. Waiting Time of MBML

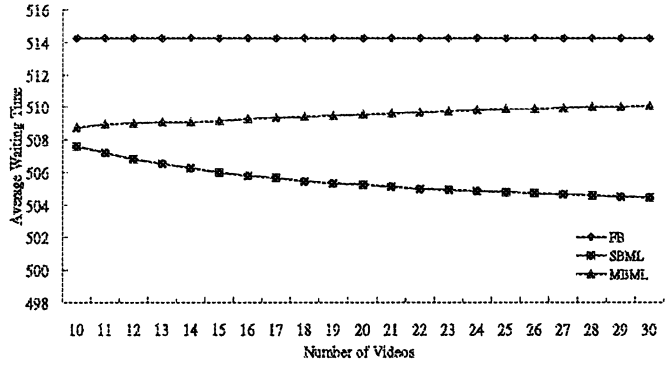
Consider any time slot. Let B be the set of m videos such that there are no new-coming viewers for them during the previous



(a)

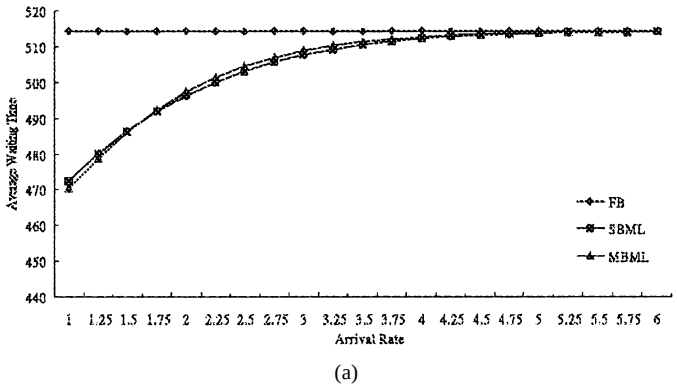


(b)

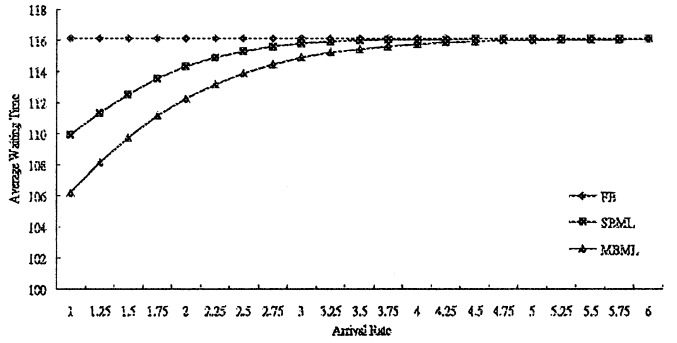


(c)

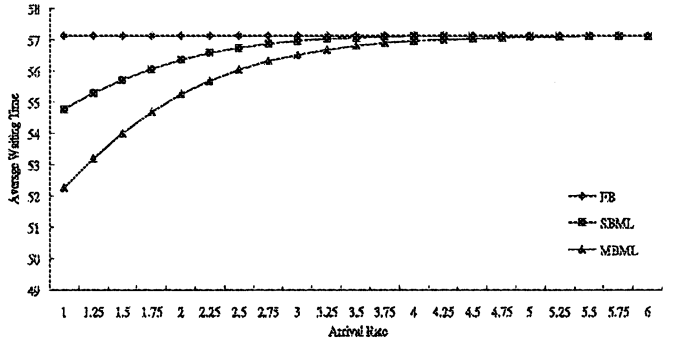
Fig. 7. Average waiting time (seconds) vs. number of videos when the arrival parameter is set to: (a) $\alpha = 1$, (b) $\alpha = 2$, and (c) $\alpha = 3$.



(a)



(b)



(c)

Fig. 8. Average waiting time (in seconds) vs. arrival parameter when the number of channels per video is set to: (a) 3, (b) 5, and (c) 6.

time slot. Still, this probability is $NN(B)$. In MBML, we will use m empty slots of videos in B to help the broadcasting of all other $n - m$ videos with viewers. So the service time is $W'_0 = (n-m)\delta/n$. Then, we are likely to serve the new-coming viewers for V_1 first, followed by V_2, then V_3, etc. So the service time for V_i will be

$$W'_j = \frac{\delta + (k-1)\sum_{c=0}^{j-1} W'_c}{n}.$$

Let's write the accumulated time $T'_c = \sum_{j=0\cdots c} W'_j$. It can be derived that

$$T'_c = \frac{(n-m)(n+k-1)^c\delta}{n^{c+1}} + \frac{((n+k-1)^c - (n)^c)\delta}{(k-1)n^c}.$$

Following similar derivation for SBML, given B, the number of videos that can receive the borrow-and-return service in the current time slot, denoted by $p(B)$, must satisfy $T'_{p(B)} \le \delta < T'_{p(B)+1}$. With $p(B)$, we can derive the aggregated amount of saving in waiting time, which is the same as (1). Then, taking different combinations of B into consideration, we have the average viewer waiting time

$$T_{MBML} = \frac{\sum_{j=1}^{n} \lambda_j\delta \cdot \delta/2 - \sum_{B \subseteq \{V_1, \ldots V_n\}} NN(B) \cdot SA(B)}{\sum_{j=1}^{n} \lambda_j\delta}.$$

C. Comparison

Based on the above analyses, we compare the average waiting time incurred by the original FB scheme against our SBML and MBML when applied to FB. In the first experiment, we vary the

number of videos n between 10 and 30. Each video is of length 120 minutes and is supported by 3 channels. Videos' arrival rates are determined by an arrival parameter called α, where α is a positive value. Specifically, the first video V_1 has a user arrival rate of α per video segment time, the second video V_2 has a user arrival rate of $\alpha - (\alpha/2(n-1))$, the third video V_3 has a user arrival rate of $\alpha - 2\times(\alpha/2(n-1))$, etc. In general, the ith video V_i has a user arrival rate of $\alpha - (i-1)\times(\alpha/2(n-1))$, and the last video V_n has a user arrival rate of $\alpha/2$. The video segment time in FB is the video length divided by $2^k - 1$, where k is the number of channels assigned to the video. Intuitively, the arrival rate drops by $\alpha/2(n-1)$ for each video. We do this purposely so as to vary the arrival rates of videos. The results are shown in Fig. 7, which indicates that SBML performs the best, which is followed by MBML, and then by FB.

In the next experiment, we vary the arrival parameter α between 1.0 and 6.0. There are 20 videos. Each video is still of length 120 minutes, but can be supported by 3, 5, or 6 channels. The results are shown in Fig. 8. As can be seen, as there are more channels per video, MBML will eventually outperform SBML. This is probably because MBML can better utilize the borrowed capacity. Also, the amount of improvement over FB will decrease as the arrival parameter increases. This is reasonable because as the load becomes higher, less borrow-and-return activities may happen. So our result is more useful when users' requests fluctuate dynamically.

V. Conclusions

The video broadcasting service is already popular in CATV systems. Asynchronous video service is likely to grow quickly when the network infrastructure is ready. In this paper, we have proposed a general borrow-and-return model that can be applied immediately to any broadcasting scheme that has the FSFC property to further reduce viewer's waiting time. Indeed, many well-know broadcasting schemes share the FSFC property and thus may enjoy the SBML and MBML schemes proposed in this work to further reduce viewer's waiting time. Analyses and comparisons are provided to justify the effectiveness of this approach.

References

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On optimal batching policies for video-on-demand storage servers," in *IEEE Proceedings of the International Conference on Multimedia Computing and Systems*, June 1996, pp. 253–258.

[2] ——, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," in *IEEE Proceedings of the International Conference on Multimedia Computing and Systems*, June 1996, pp. 118–126.

[3] P. W. Agnew and A. S. Kellerman, *Distributed Multimedia*: Addison Wesley.

[4] S. W. Carter and D. D. E. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. of the International Conference on Computer Communications and Networks*, September 1997, pp. 200–207.

[5] Y. H. Chang, D. Coggins, D. Pitt, and D. Skellern, "An open-system approach to video on demand," *IEEE Communication Magazine*, vol. 32, pp. 68–80, May 1994.

[6] T. Chiueh and C. Lu, "A periodic broadcasting approach to video-on-demand service," *International Society for Optical Engineering*, vol. 2615, pp. 162–169, October 1995.

[7] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *Proc. of ACM Multimedia*, 1994, pp. 15–23.

[8] ——, "Dynamic batching policies for an on-demand video server," *Multimedia Systems*, vol. 4, no. 3, pp. 112–121, June 1996.

[9] D. Deloddere, W. Verbiest, and H. Verhille, "Interactive video on demand," *IEEE Communications Magazine*, vol. 32, pp. 82–88, May 1994.

[10] L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popolar videos," in *International Workshop on Network and Operaing Systems Support for Digital Audio and Video*, August 1998, pp. 317–329.

[11] L. Gao and D. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," *IEEE Multimedia*, pp. 117–121, 1999.

[12] W. Hodges, S. Mabon, and J. T. , Jr., "Video on demand: Architecture, systems, and applications," *Building an Infrastructure for Managing Compressed Video Systems*, pp. 791–803, 1993.

[13] K. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *ACM Multimedia*, September 1998, pp. 191–200.

[14] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *ACM SIGCOMM*, September 1997.

[15] L.-S. Juhn and L.-M. Tseng, "Fast broadcasting for hot video access," *Real-Time Computing Systems and Applications*, pp. 237–243, October 1997.

[16] ——, "Harmonic broadcasting for video-on-demand service," *IEEE Transactions on Broadcasting*, vol. 43, no. 3, pp. 268–271, September 1997.

[17] ——, "Enhanced harmonic data broadcasting and receiving scheme for popular video service," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 2, pp. 343–346, May 1998.

[18] ——, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Transactions on Broadcasting*, vol. 44, no. 1, pp. 100–105, March 1998.

[19] T. L. Kunii *et al.*, "Issues in storage and retrieval of multimedia data," *Multimedia Systems*, vol. 3, no. 5, pp. 298–304, 1995.

[20] T. D. C. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, vol. 1, no. 3, pp. 14–24, March 1994.

[21] B. Ozden, R. Rastogi, and A. Silberschatz, "On the design of a low cost video-on-demand storage system," *Multimedia Systems*, vol. 4, no. 1, pp. 40–54, 1996.

[22] J.-F. Paris, "A simple low-bandwidth broadcasting protocol," in *International Conference on Computer Communication and Network*, 1999, pp. 118–123.

[23] J.-F. Paris, S.-W. Carter, and D.-D. Long, "Efficient broadcasting protocols for video on demand," *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 127–132, July 1998.

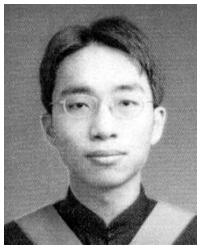
[24] ——, "A hybrid broadcasting protocol for video on demand," in *Multimedia Computing and Networking Conference*, 1999, pp. 317–326.

[25] W. D. Sincoskie, "System architecture for a large scale video on demand service," *Computer Networks and ISDN Systems*, vol. 22, pp. 565–570, 1991.

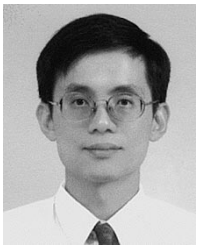
[26] Y.-C. Tseng, M.-H. Yang, and C.-H. Chang, "A recursive frequency-splitting scheme for broadcasting hot videos in vod servic," *IEEE Transactions on Communications*, vol. 50, no. 8, pp. 1348–1355, August 2002.

[27] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *IEEE Multimedia Systems*, vol. 4, pp. 197–208, 1996.

Ming-Hour Yang received the M.S. degree in Electronic Engineering from the Chung-Hua University, Taiwan, in 1996, and the Ph.D. degree in Computer Science and Information Engineering from the National Central University, Taiwan, in 2001. He was a Research Fellow of National Strategic Studies Institute at National Defense University, Taiwan. He is currently an Assistant Professor of the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan. His interests include video on demand, parallel and distributed computing, wireless network, and network security.



Chi-He Chang received the B.S. degree in Computer Science from the Chung Yuan Christian University, Chung-Li, Taiwan, R.O.C., in 1998, and the M.S. degree in Computer Science and Information Engineering from National Central University, Chung-Li, Taiwan, R.O.C. in 2000. Since 2002, he is a Ph.D. student in the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan. His interests include video on demand, interactive multimedia and wireless network.



Yu-Chee Tseng received the B.S. and M.S. degrees in Computer Science from the National Taiwan University and the National Tsing-Hua University in 1985 and 1987, respectively. He worked for the D-LINK Inc. as an Engineer in 1990. He obtained the Ph.D. in Computer and Information Science from the Ohio State University in January of 1994. From 1994 to 1996, he was an Associate Professor at the Department of Computer Science, Chung-Hua University. He joined the Department of Computer Science and Information Engineering, National Central University in 1996, and has become a Full Professor since 1999. Since August 2000, he has become a Full Professor at the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan. Dr. Tseng has served as a Program Committee Member in many international conferences, as a Program Chair in the *Wireless Networks and Mobile Computing Workshop*, in years 2000 and 2001, as an Associate Editor in *The Computer Journal*, and as a Guest Editor in *ACM Wireless Networks*, IEEE TRANSACTIONS ON COMPUTERS, *Journal of Internet Technology*, and *Wireless Communications and Mobile Computing*. His research interests include wireless communication, network security, parallel and distributed computing, and computer architecture. Dr. Tseng is a member of the IEEE Computer Society.