Data Broadcasting and Seamless Channel Transition for Highly-Demanded Videos

Yu-Chee Tseng*, Chi-Ming Hsieh, Ming-Hour Yang, Wen-Hwa Liao, and Jang-Ping Sheu
Department of Computer Science and Information Engineering
National Central University
Chung-Li, 32054, Taiwan
*Email: yctseng@csie.ncu.edu.tw (corresponding author)

Abstract—**One way to broadcast a popular video is to let multiple users share fewer channels. The stress on channel demand can be alleviated without sacrificing viewers' waiting time. One such scheme that interests us is the Fast Broadcasting (FB) scheme [8], [11], which can broadcast a popular video using k channels without keeping new-coming viewers waiting for more than $O(D/2^k)$ time, where D is the length of the video. In this paper, we propose two enhancements to the FB scheme. First, since the level of demand on a video may change by time, we show how to dynamically change the number of channels assigned to the video and seamlessly perform this transition. Clients currently viewing this video will not experience any disruption during the transition. Second, given a set of channels and a set of popular videos, we propose a scheme to assign these channels to the videos such that the average viewers' waiting time is minimal. From the system manager's point of view, these enhancements will make the FB scheme more attractive.**

Keywords: Broadcasting, Cable TV, Channel Allocation, Digital Video Broadcasting, Video-On-Demand (VOD)

I. Introduction

With the advancement of broadband networking technology and the growth of processor speed and disk capacity, VOD (video-on-demand) services have become possible [12], [14]. 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 [3], [7], [15]. Depending on how channels are utilized, the system can be classified as *interactive* [6], [13] or *broadcasting* [2], [5]. An interactive system serves each client with a dedicated channel and emulates the VCR functions (e.g., forward, rewind, pause, search, etc.). Apparently, such systems could easily run out of channels because the growth in the number of channels can never keep up with the growth in the number of clients. On the contrary, to relieve the stress on channel demand, a broadcasting system tries to serve multiple clients with less channels shared by the clients. Such an approach is more appropriate for popular or hot videos that may interest many viewers at a certain period of time. Due to its sharing nature, some VCR functions may be sacrificed. However, efforts should be made to provide a near-VOD quality of service.

This work is supported by the National Science Council of the Republic of China under Grant # NSC89-2213-E-008-023 and #NSC89-2213-E-008-025.

In this paper, we consider the broadcasting approach. One important issue in such a system is the *data scheduling problem*, which refers to how a video server arranges video data on a given bandwidth and how a client grabs the data streams to play the required video. Considerations in this problem include how to reduce a new-coming viewer's waiting time before he/she can start the service, and how to reduce the buffering space requirement at the client side.

To reduce the new-coming viewers' waiting time, reference [4] proposes a periodic broadcasting approach, which can decrease the maximum waiting time linearly with respect to the number of channels assigned to a video. In [1], [16], a scheme called *pyramid* is proposed, which can reduce the maximum waiting time in an exponential ratio with respect to the number of channels used. However, this requires buffering portions of the video on the client side. In [8], [11], a scheme called *Fast Broadcasting (FB)* is proposed, which can further reduce the waiting time and the buffering requirements. In [9], [10], a broadcasting scheme derived based on the concept of *harmonic series* is proposed; the scheme allows the bandwidth assigned to a video not equal to a multiple of the bandwidth of one channel.

The above schemes are all aimed at reducing the waiting time given a *fixed* bandwidth for *one* video. In this paper, we observe two deficiencies associated with these schemes. First, the level of demand on a video will be changed by many factors (such as day of a week, time of a day, or social events). The bandwidth received by a video should, to a certain degree, reflect the "level of hotness" of the video. None of the above reviewed schemes can adapt to this change. It will be desirable to dynamically adjust the bandwidth assigned to a video on-the-fly in a seamless manner. Second, from a system manager's viewpoint, given some bandwidth and a set of popular videos, it is not clear how much bandwidth should be assigned to each video to make the most benefit from the broadcasting service. This is not addressed in existing work either.

This paper is motivated by the above observations. In this paper, we choose the FB scheme [8], [11] as our target and propose two enhancements to it. First, we design a *channel transition* scheme on top of the FB scheme, so that the number of channels assigned to a video can be adjusted on-the-fly seamlessly, in the sense that while the video is undergoing some change on the number of channels used, the clients currently viewing this video will not experience any disruption. Second, we propose a greedy scheme to assign a set of channels to a set of videos, given the request arrival rates of these videos. The goal is to reduce the average waiting time incurred on all clients to start their video services. Our scheme guarantees the average waiting time to be minimal. These enhancements will certainly make the FB scheme more attractive.

The rest of this paper is organized as follows. In Section II, we give some background of the FB scheme and then formally define the problem to be solved in this paper. Our two enhancements to the FB scheme are presented in Section III and Section IV, respectively. Conclusions are drawn in Section V.

II. Backgrounds and Motivations

A. *Review of the FB Scheme*

Below, we review the FB scheme proposed by [8], [11]. Consider a video V of length D with consumption rate 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 solve this problem, the FB scheme assumes that k channels, $C_0, C_1, \ldots, C_{k-1}$, each of bandwidth b, are assigned to V. These channels will work together to broadcast V with some special arrangement. The major goal is to reduce the new-coming viewers' waiting time before it can start the service of V.

The video server should use the following rules to broadcast V:

1) Partition V evenly into N data 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 use $\circ$ as the concatenation operator). The length of each segment is $\delta = D/N = D/(2^k - 1)$.

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 shown in Fig. 1. Channel C_0 broadcasts the first segment, S_1, periodically, C_1 broadcasts the next two segments, S_2, and S_3, periodically, C_2 broadcasts the next four segments, S_4, S_5, S_6, and S_7, periodically, etc. Note that none of these channels broadcasts the complete video V.

To view V, a client should monitor and receive data from all k channels 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 data segments starting from the first time slot.

3) Right at the moment when step 2 begins, start to consume the video $S_1 \circ S_2 \circ \cdots \circ S_N$.

Let's use an example to show how the FB scheme works. In Fig. 1, suppose the video server allocates $k = 4$ channels to V. So V will be partitioned evenly into $N = 2^4 - 1 = 15$ segments. For a client starting at time t_0 in Fig. 1, in the first time slot, it will receive segments S_1, S_2, S_4, S_8 from C_0, C_1, C_2, C_3, respectively. 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 $t_0 + N\delta = t_0 + 15\delta$. The reader should be able to derive similar results easily for viewers starting from any other time slot.

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 viewers' maximum waiting time. A client has to wait no more 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 the number of channels can reduce the maximum 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.

What the client has to pay for is the extra buffer space for those premature segments. This will depend on the time slot when a client begins its service. The maximum buffer requirement is less than half of the video size, $Db/2$ [8]. In some cases, it is possible for a client to play the video without buffering if it is willing to wait longer. To see this, consider Fig. 1. A client starting at time t_1 does not need to buffer any segment because 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.

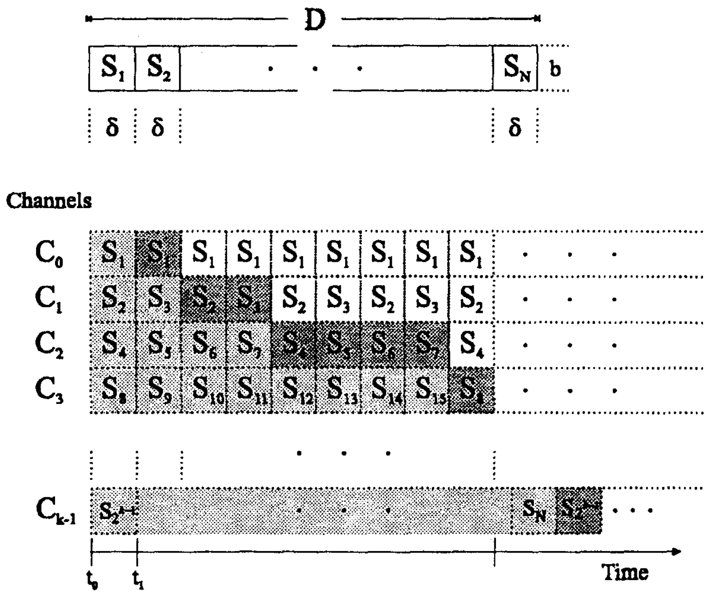
Fig. 1. Data scheduling of the Fast Broadcasting (FB) scheme.

B. Problems with the FB Scheme

We observe two deficiencies associated with the FB scheme. First, the FB scheme only considers the data scheduling of a video given a *fixed* number of channels. Since the level of demand on the video may change by time, it is desirable that the channel assignment can adapt to such changes. We call it *channel transition* when the video is undergoing some change on the number of channels used. One trivial solution is to allocate a new set of channels for the new configuration to serve new-coming users, while keeping the existing set of channels unchanged to serve existing users. The existing set of channels can be released after all existing users have received their required video segments. Apparently, this wastes the communication bandwidth. More seriously, it requires some spare channels to perform the transition. In extreme cases, the following *deadlock* scenario may occur: when the server has run out of channels, the system may not be able to perform any transition because there are no enough spare channels to do so, unless the server stops the service of some videos to vacate some channels, which is very undesirable.

It is desirable that we can dynamically adjust the number of channels assigned to a video on-the-fly. The transition should be *seamless*, in the sense that during the transition period, clients currently viewing the video will not experience any disruption. In the meanwhile, new clients may keep on coming, and their service can be started right away.

Second, the FB scheme considers only one video at a time. From a system manager's viewpoint, given a set of popular videos each with a certain level of demand, it is not clear how many channels should be assigned to each video to make the most benefit from the broadcasting service. We formulate this problem as follows. Suppose that the system has totally K channels to support m popular videos $V_1, V_2, \ldots, V_m$ of lengths $D_1, D_2, \ldots, D_m$, respectively. Let λ_i be the request arrival rate of $V_i, i = 1, \ldots, m$. Also, let k_i be the number of channels assigned to V_i. According to the FB scheme, the average waiting time for V_i is $D_i/(2 \cdot (2^{k_i} - 1))$. Considering all videos, our goal should be to minimize the average waiting time of all clients, i.e.,

$$\text{Minimize} \quad \sum_{i=1}^{m} \lambda_i * \frac{D_i}{2 \cdot (2^{k_i} - 1)}, \tag{1}$$

subject to

$$\sum_{i=1}^{m} k_i \leq K, \quad \text{such that} \quad k_i \geq \alpha,$$

where α is a small integer indicating the minimum number of channels to be assigned to each video. Since our basic assumption is that all videos are highly-demanded videos, a reasonable value is $\alpha \geq 3$ or 4 (but our scheme will work with any fixed α). For instance, for a typical 120-minute video, $\alpha = 3$ gives an average waiting time of $120/(2 \cdot (2^3 - 1)) \approx 8.57$ minutes, and $\alpha = 4$ an average waiting time of $120/(2 \cdot (2^4 - 1)) = 4$ minutes. One obvious simplification is to change the constraint to $\sum_{i=1}^{m} k_i = K$, since it is always beneficial if we can increase the number of channels assigned to a video.

III. Channel Transition for the FB Scheme

In this section, we will show how to modify the FB scheme to make seamless channel transition possible. We consider one video V of length D. First, V will be padded with some dummy video stream at its end before being broadcast (Section III-A). Then, we will establish the relationship of channel contents at different channel assignments. In particular, we will identify a special case that will lead to a "good" relationship (Section III-B). Based on the relationship, we will show how to seamlessly change the number of channels assigned to a video (Section III-C).

A. Data Padding

Recall the original FB scheme. Given, for instance, 3 and 4 channels to V, the length of each segment will be $D/7$ and $D/15$, respectively. Since the denominators are mutually prime, there is no clear correspondence relationship between the segments obtained from these two channel assignments. This is similar for most other assignments. Below, we will perform some padding on V to resolve this problem.

Recall that there is a minimum requirement that V has at least α channels. At the end of V, a dummy video stream V_{dummy} of length

$$|V_{dummy}| = \frac{D}{2^\alpha - 1}$$

is padded. Now let

$$\begin{aligned} V' &= V \circ V_{dummy} \\ D' &= |V \circ V_{dummy}| = \frac{2^\alpha}{2^\alpha - 1} \cdot D. \end{aligned} \qquad (2)$$

Suppose that k channels $C_0, C_1, \ldots, C_{k-1}$ are assigned to V ($k \geq \alpha$). We modify the rules used by the video server as follows.

1. Partition V' evenly into N data segments, $S_1, S_2, \ldots, S_N$, where $N = 2^k$, i.e., $S_1 \circ S_2 \circ \cdots \circ S_N = V'$. Denote by $\delta' = D'/N = D'/2^k$ the length of each segment.
2. Divide each channel C_i, $i = 0, \ldots, k-1$, into time slots of length δ'. On C_i, the video server broadcasts 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 to the same time slot.

Note that in step 1, the definition of N is slightly different from that in the original FB scheme (which assigns $N = 2^k - 1$). Step 2 looks exactly the same as the original FB scheme, but now we are broadcasting V' instead of V. One subtlety here is that the last segment S_N is never broadcast on any of the channels. However, this does not matter because S_N always contains dummy data:

$$|S_N| = \frac{D'}{2^k} \leq \frac{D'}{2^\alpha} = \frac{D}{2^\alpha - 1} = |V_{dummy}|.$$

On the client side, it performs exactly the same action as the original FB scheme. But now the video content is $S_1 \circ S_2 \circ \cdots \circ S_{N-1}$, which contains a complete version of V. Note that the last $2^{k-\alpha} - 1$ segments contain dummy videos. One possible application of these dummy segments is to play advertisements or previews of popular videos.

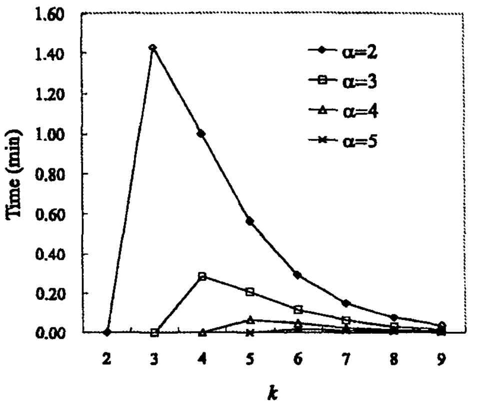

Fig. 2. The increase in average waiting time after data padding for a 120-minute video using k channels.

Recall that in the original FB scheme, the length of each time slot is $\delta = D/(2^k - 1)$. After the above padding, the length of each time slot is increased by

$$\begin{aligned} \delta' - \delta &= \frac{D'}{2^k} - \frac{D}{2^k - 1} \\ &= \frac{2^k - 2^\alpha}{(2^\alpha - 1) \cdot 2^k \cdot (2^k - 1)} \cdot D. \end{aligned}$$

The average waiting time will be increased by $(\delta' - \delta)/2$. Fortunately, this value quickly converges to 0 as k increases. For instance, if $\alpha = 2$, for a 120-minute video, the average waiting time is increased by 1.43 minutes when $k = 3$, by 1.00 minute when $k = 4$, and by 0.56 minute when $k = 5$. This increase is insignificant. If α is larger, the increase is much less. In Fig. 2, we show the increase of the average waiting time at various values of α and k for a 120-minute video.

After the padding, one important property is that we can easily establish the relationship of segments associated with different channel assignments. To formulate the relationship, let's denote by $S_1^k, S_2^k, \ldots, S_N^k$ the N segments of V' when k channels are used. For instance, assuming $\alpha = 2$, we show the segments of V' at different values of k in Fig. 3. A segment at a smaller k is always equal to the concatenation of some segments at a larger k (e.g., $S_1^2 = S_1^3 \circ S_2^3 = S_1^4 \circ S_2^4 \circ S_3^4 \circ S_4^4$). Such a property is crucial to our yet-to-be-presented channel transition scheme. However, no such relationship can be established in the original FB scheme. The reason is simple: the FB scheme partitions V into $2^k - 1$ segments, whereas ours partitions V' into 2^k segments.

Lemma 1: Let p and q be two integers, $p > q \geq \alpha$. For any i, $1 \leq i \leq 2^q$, we have

$$S_i^q = S_{(i-1)r+1}^p \circ S_{(i-1)r+2}^p \circ \cdots \circ S_{ir}^p,$$

where $r = 2^{p-q}$.

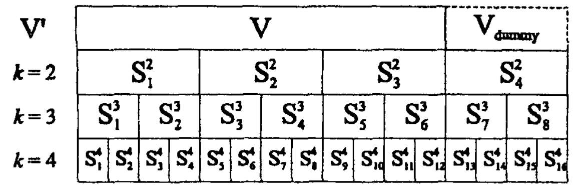
V'
V
V_dummy
k=2
S²₁ S²₂ S²₃ S²₄
k=3
S³₁ S³₂ S³₃ S³₄ S³₅ S³₆ S³₇ S³₈
k=4
S⁴₁ S⁴₂ S⁴₃ S⁴₄ S⁴₅ S⁴₆ S⁴₇ S⁴₈ S⁴₉ S⁴₁₀ S⁴₁₁ S⁴₁₂ S⁴₁₃ S⁴₁₄ S⁴₁₅ S⁴₁₆

Fig. 3. The relationship of segments when k (= 2, 3, 4) channels are assigned to a video V. It is assumed that $\alpha = 2$, so $|V_{dummy}| = D/3$.

B. Relationship of Channel Contents

In this section, we will first model the channel contents given a certain number of channels to a popular video. Then we will raise a special case which reveals some nice properties that can facilitate channel transition.

We will still use the video V, and the same notations in Section III-A (such as V', D', k, S_i^k, $C_0, C_1, \ldots, C_{k-1}$), in our discussion. For ease of presentation, we assume that the system starts at time 0. The content of a channel can be regarded as an infinite video stream with respect to the time axis starting from time 0. Supposing that Ψ represents a channel content, we will denote by $\Psi(t)$ the content of Ψ at time point t. To denote the special case that no content (or no meaningful content) is broadcast at time t, we will write $\Psi(t) = \emptyset$. Also, the video content of Ψ in time interval $[t_1, t_2]$ will be written as $\Psi(t_1, t_2)$.

Definition 1: Given an integer k and an integer d such that $k \geq \alpha$ and $0 \leq d < k$, we define the infinite video stream

$$T_d^k = cycle(S_{2^d}^k \circ S_{2^d+1}^k \circ \cdots \circ S_{2^{d+1}-1}^k),$$

where $cycle(s)$ means an infinite repetition of the given video stream s (i.e., $cycle(s) = s \circ s \circ s \circ \cdots$). Also, define $\tilde{T}^k$ to be a set of k video streams such that

$$\tilde{T}^k = \{T_d^k, d = 0..k-1\}.$$

Intuitively, T_d^k is the video content broadcast on channel C_d by our modified FB scheme, and $\tilde{T}^k$ is the collection of video contents broadcast on channels $C_0, C_1, \ldots, C_{k-1}$. Next, we will change the channel content T_d^k by performing a shift operation.

Definition 2: Given an integer k and an integer d such that $k \geq \alpha$ and $0 \leq d < k$, we define the infinite video stream

$$U_d^k = cycle(shift(2^{k-\alpha} - 1, S_{2^d}^k \circ S_{2^d+1}^k \circ \cdots \circ S_{2^{d+1}-1}^k)),$$

where $shift(i, s_1 \circ s_2 \circ \ldots)$ is a function which cyclically rotates the parameters, $s_1, s_2, \ldots$, to the right by i times. Also, define $\tilde{U}^k$ to be a set of k video streams such that

$$\tilde{U}^k = \{U_0^k, U_1^k, \ldots, U_{k-1}^k\}.$$

Intuitively, U_d^k is obtained from T_d^k by cyclically shifting the latter's video segments to the right by $2^{k-\alpha} - 1$ time slots. Fig. 4 illustrates the relationship between T_d^k and U_d^k when $\alpha = 2$ and $k = 2, 3, 4$. When $k = 2$, we have $T_d^2 = U_d^2$ since there is no shift being taken ($2^{k-\alpha} - 1 = 0$). When $k = 3$, the video is shifted by $2^{k-\alpha} - 1 = 1$ slot, and when $k = 4$, the video is shifted by $2^{k-\alpha} - 1 = 3$ slots. To summarize, let's compare Fig. 4(a) and Fig. 4(b). Before the shifting, the *starting time* of the first segment in each channel (i.e., $\{S_1^2, S_2^2\}, \{S_1^3, S_2^3, S_3^3\}$, and $\{S_1^4, S_2^4, S_4^4, S_8^4\}$) is aligned (at time 0). While, after the shifting, the *ending time* of these segments are aligned (at time t_1).

Observe that for each individual $\tilde{U}^k$, all the channel contents $\{U_0^k, U_1^k, \ldots, U_{k-1}^k\}$ are shifted by the same amount of time. Since our scheme allows a new user to start at the beginning of any time slot, each individual $\tilde{U}^k$ still describes how to broadcast video V' to its viewers when k channels are used. The most important effect we want from the shifting is a corresponding relationship between two channel configurations $\tilde{U}^k$ and $\tilde{U}^{k+1}$, which is derived in the following.

For each U_d^k, time is slotted by length of $D'/2^k$. Let's denote by $U_d^k[j]$ the video content of U_D^k at the j-th time slot. Note that here we will count j starting from 0, i.e., $U_d^k[0]$ is the video content at the first time slot, $U_d^k[1]$ that at the second time slot, etc. The following lemma derives the content of $U_d^k[j]$ for an arbitrary j.

Lemma 2: For any $j \geq 0$, we have

$$U_d^k[j] = S_{((j-(2^{k-\alpha}-1)) \bmod 2^d)+2^d}^k.$$

Proof: We show how the subscript of S is derived. The "$2^{k-\alpha} - 1$" is for the shifting effect. The "mod 2^d" is because the channel has a cycle length of 2^d slots. The final part of "$+2^d$" is for the offset (U_d^k contains segments $S_{2^d}^k, S_{2^d+1}^k, S_{2^d+2}^k, \ldots$). □

Theorem 1: For any j, we have

$$U_d^k[j] = \begin{cases} U_d^{k+1}[2j] \circ U_{d+1}^{k+1}[2j+1] \\ \qquad \text{if } j - (2^{k-\alpha} - 1) \bmod 2^d = 0 \\ U_{d+1}^{k+1}[2j] \circ U_{d+1}^{k+1}[2j+1] \\ \qquad \text{if } j - (2^{k-\alpha} - 1) \bmod 2^d \neq 0 \end{cases}.$$

Proof: By Lemma 1 and Lemma 2, we have

$$\begin{aligned} U_d^k[j] &= S_{((j-(2^{k-\alpha}-1)) \bmod 2^d)+2^d}^k \\ &= S_{2(((j-(2^{k-\alpha}-1)) \bmod 2^d)+2^d)-1}^{k+1} \\ &\quad \circ S_{2(((j-(2^{k-\alpha}-1)) \bmod 2^d)+2^d)}^{k+1} \\ &= S_{((2j-2^{k+1-\alpha}+2) \bmod 2^{d+1})+2^{d+1}-1}^{k+1} \\ &\quad \circ S_{((2j-2^{k+1-\alpha}+2) \bmod 2^{d+1})+2^{d+1}}^{k+1}. \end{aligned} \tag{3}$$

Note that the last equivalence is from the property of modular arithmetic. First, we prove the case of $j - (2^{k-\alpha} - 1) \bmod 2^d = 0$. This simplifies Eq. (3) to

$$U_d^k[j] = S_{2^{d+1}-1}^{k+1} \circ S_{2^{d+1}}^{k+1}. \tag{4}$$

By Lemma 2, we can derive that

$$\begin{aligned} U_d^{k+1}[2j] &= S_{((2j-(2^{k+1-\alpha}-1)) \bmod 2^d)+2^d}^{k+1} \\ &= S_{((2(2^{k-\alpha}-1)-(2^{k+1-\alpha}-1)) \bmod 2^d)+2^d}^{k+1} \\ &= S_{(-1 \bmod 2^d)+2^d}^{k+1} \\ &= S_{2^{d+1}-1}^{k+1} \end{aligned}$$

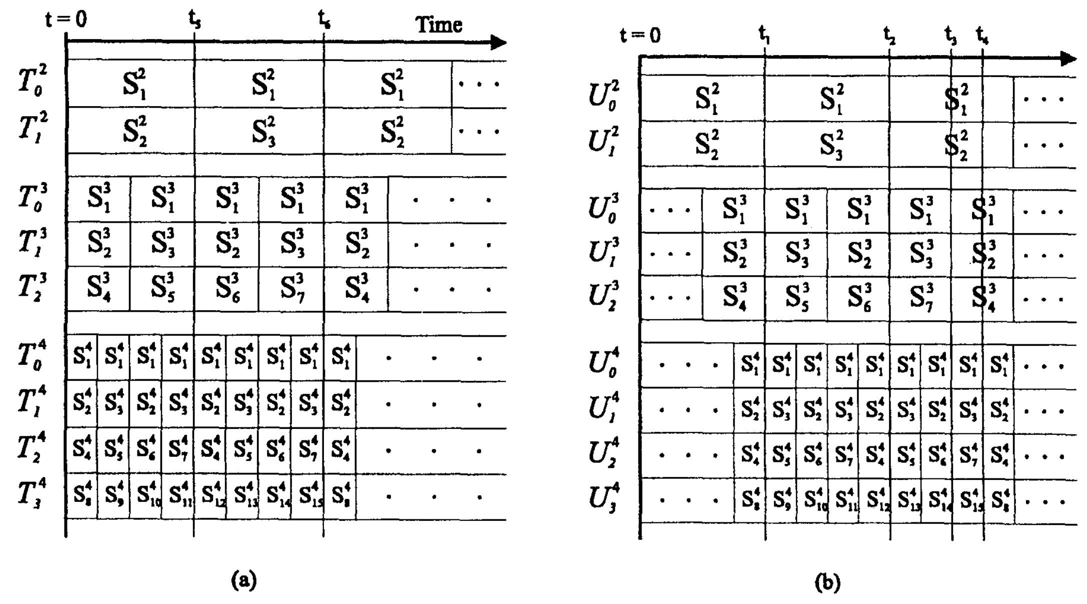

Fig. 4. (a) Channel contents $\tilde{T}^k$ at $k = 2, 3, 4$. (b) Shifted channel contents $\tilde{U}^k$ at $k = 2, 3, 4$. For clarity, the first few segments of $\tilde{U}^k$ are not shown so as to observe the shifted effect.

and that

$$\begin{aligned} U_{d+1}^{k+1}[2j+1] =& S^{k+1}_{((2j+1-(2^{k+1-\alpha}-1))\bmod 2^{d+1})+2^{d+1}} \\ =& S^{k+1}_{((2(2^{k-\alpha}-1)+1-(2^{k+1-\alpha}-1))\bmod 2^{d+1})+2^{d+1}} \\ =& S^{k+1}_{(0 \bmod 2^{d+1})+2^{d+1}} \\ =& S^{k+1}_{2^{d+1}}. \end{aligned}$$

So this case has been proved.

For the case of $j - (2^{k-\alpha} - 1) \bmod 2^d \neq 0$, we derive from Eq. (3) that

$$\begin{aligned} U_d^k[j] = & S^{k+1}_{((2j-2^{k+1-\alpha}+1)\bmod 2^{d+1})+2^{d+1}+1-1} \\ & \circ S^{k+1}_{((2j+1-2^{k+1-\alpha}+1)\bmod 2^{d+1})+2^{d+1}}. \end{aligned} \tag{5}$$

The parameter on the left-hand side of $\circ$ is so written because the "mod" operation will result in a non-zero value, making possible to move a value of 1 outside the "mod" operator. Then by Lemma 2 this becomes $U_{d+1}^{k+1}[2j]$. The parameter on the right-hand side of $\circ$ is obtained from trivial arithmetic. It then can be translated by Lemma 2 to $U_{d+1}^{k+1}[2j+1]$. So this completes the proof. □

Intuitively, the above theorem states that the content of U_d^k at the j-th time slot is the concatenation of some contents of U_d^{k+1} and U_{d+1}^{k+1} at the $2j$-th and $(2j+1)$-th time slots. Note that the length of a time slot in the former equals exactly two times that of the latter. In fact, the j-th time slot of $\tilde{U}^k$ is exactly the $2j$-th and $(2j+1)$-th time slots of $\tilde{U}^{k+1}$ (recall that we count j starting from 0). Thus, what is broadcast in $\tilde{U}^k$ must be concurrently broadcast in $\tilde{U}^{k+1}$.

Corollary 1: At any point of time t, the relation holds:

$$\begin{aligned} &\{U_0^k(t), U_1^k(t), \ldots, U_{k-1}^k(t)\} \\ &\subseteq \{U_0^{k+1}(t), U_1^{k+1}(t), \ldots, U_k^{k+1}(t)\}. \end{aligned} \tag{6}$$

For any d, $0 \leq d < k$, the video content $U_d^k(t)$ can be found in either $U_d^{k+1}(t)$ or $U_{d+1}^{k+1}(t)$.

For instance, in Fig. 4(b), in time interval $[t_1, t_2]$, segments S_1^2 and S_3^2 of $\tilde{U}^2$ can be found in S_1^3, S_2^3, S_5^3, and S_6^3 of $\tilde{U}^3$, which in turn can be found in $S_1^4, S_2^4, S_3^4, S_4^4, S_9^4, S_{10}^4, S_{11}^4$, and S_{12}^4 of $\tilde{U}^4$ (by Lemma 1, $S_1^2 = S_1^3 \circ S_2^3 = S_1^4 \circ S_2^4 \circ S_3^4 \circ S_4^4$ and $S_3^2 = S_5^3 \circ S_6^3 = S_9^4 \circ S_{10}^4 \circ S_{11}^4 \circ S_{12}^4$). Similarly, the relationship can be found for the video content broadcast in time interval $[t_3, t_4]$. On the contrary, such a containment relationship does not exist in the example shown in Fig. 4(a). For instance, segment S_3^2 of $\tilde{T}^2$ is broadcast in time interval $[t_5, t_6]$, but the same content is neither broadcast on $\tilde{T}^3$ nor on $\tilde{T}^4$.

The following corollary can be easily extended from the previous corollary.

Corollary 2: For any two integers p and q, $p > q \geq \alpha$, at any point of time t, the relation holds:

$$\{U_0^q(t), U_1^q(t), \ldots, U_{q-1}^q(t)\} \subseteq \{U_0^p(t), U_1^p(t), \ldots, U_{p-1}^p(t)\}.$$

C. Seamless Channel Transition

In this section, we assume that video V is currently supported by k channels. We will show how to seamlessly change the number of channels from k to k' (both k and $k' \geq \alpha$). We call this a *channel transition.* Let $\Delta k = k' - k$. If $\Delta k > 0$, this is called a *positive channel transition (PCT).* If $\Delta k < 0$, this is called a *negative channel transition (NCT).*

C.1 PCT

The fundamental of PCT is Corollary 2. Suppose that the server is currently using $\tilde{U}^k$ to broadcast V. Corollary 2 states that at any time t, the server can stop using $\tilde{U}^k$ and transit to $\tilde{U}^{k'}$ to broadcast V. A client currently in the system will not miss any content that it expects to receive in the future after the transition. However, in reality, the transition time can be chosen at the beginning of any time slot of $\tilde{U}^{k'}$, since this is the only point new-coming clients can enter under the configuration $\tilde{U}^{k'}$. We summarize the server's PCT rules as follows:

1) Let the current time be t. Determine the transition time point t' to be the beginning of the nearest coming time slot in $\tilde{U}^{k'}$.

2) Starting from time t', use the new channel set to broadcast $\tilde{U}^{k'}$.

On the client side, it should be notified of this transition. For an existing client who started before time t', it should change its receiving rule to that for $\tilde{U}^{k'}$. For a new client coming at or after t', it directly follows the receiving rule for $\tilde{U}^{k'}$.

One thing unspecified is how video data should be buffered by existing clients after the transition. There are several possibilities. The first and simplest way is to use two sets of buffers, one for $\tilde{U}^k$ and the other for $\tilde{U}^{k'}$. The client keeps on consuming V from the former buffer, while at the same time storing the data currently being broadcast on the latter buffer. Whenever any data segment is missing in the former buffer, it can be found either from the latter buffer or from one of the channels. The second way is from the observation that using two sets of buffers is waste of space. In fact, if for any time t, the video content $V'(t)$ is mapped to the same buffer address no matter $\tilde{U}^k$ or $\tilde{U}^{k'}$ is used, one set of buffers will be sufficient. The reason is that there will be no conflict in buffer usage.

The third way is more computation-intensive, but can avoid a client buffering the same video content multiple times. Recall that on the old channel $U_d^k, d = 0..k-1$, a client should receive 2^d segments. After the transition, the client can compute the segments that it expects to receive from U_d^k and translate them to those segments under $\tilde{U}^{k'}$ (by Lemma 1). Then it only downloads these missing segments. Corollary 2 guarantees that this will succeed. For instance, in Fig. 5(a) and (b), we show two possible transitions from $\tilde{U}^2$ to $\tilde{U}^3$. The gray segments are what a client should download before and after the transition. Two more examples of transitions from $\tilde{U}^2$ to $\tilde{U}^4$ are shown in Fig. 5(c) and (d).

C.2 NCT

Definition 3: Given any k, define $\tilde{U}^{k+1} \ominus \tilde{U}^k$ to be an infinite video stream Ψ such that at any time t (operator \ means *set difference.*),

$$\Psi(t) = \{U_0^{k+1}(t), U_1^{k+1}(t), \ldots, U_k^{k+1}(t)\} \setminus \{U_0^k(t), U_1^k(t), \ldots, U_{k-1}^k(t)\}. \tag{7}$$

The operator $\ominus$ identifies the video content that is broadcast on $\tilde{U}^{k+1}$ but not on $\tilde{U}^k$ at any instance. The result must be an infinite video stream Ψ, because Corollary 1 guarantees that $\Psi(t)$ must be unique. An example $\tilde{U}^4 \ominus \tilde{U}^3$ is shown in Fig. 6(a).

In fact, what we need is a special case of the above definition, as shown below.

Definition 4: Let t be the beginning of any time slot in $\tilde{U}^k$. Given any k, define the video stream

$$\begin{aligned} &\tilde{U}^{k+1} \ominus_t \tilde{U}^k \\ &= \{U_0^{k+1}(t,t), U_1^{k+1}(t, t+\delta), U_2^{k+1}(t, t+3\delta), \ldots, \\ &\quad U_k^{k+1}(t, t+(2^k-1)\delta)\} \ominus \tilde{U}^k, \end{aligned} \tag{8}$$

where $\delta = D'/2^{k+1}$ is the length of a time slot in $\tilde{U}^{k+1}$.

In this definition, we restrict each channel in $\tilde{U}^{k+1}$ to some time interval. The intuition is as follows. Suppose that the video is underwent a transition from $\tilde{U}^{k+1}$ to $\tilde{U}^k$ at time t. Consider those clients who just started their service one slot before t (i.e., at time $t-\delta$). Recall that these clients has to receive 2^d segments from U_d^{k+1} for each $d = 0..k$. Thus, after the transition, these clients still miss $2^d - 1$ segments in U_d^{k+1}, i.e., the video content in $U_d^{k+1}(t, t+(2^d-1)\delta)$ (a special case is $d = 0$, which gives an empty stream $U_0^{k+1}(t,t)$; however, we still keep it in Eq. (8) for clarity). Fig. 6(b) shows an example, which is obtained from Fig. 6(a) by restricting some channel contents to some time intervals. Note that some parts of the resulting video stream may contain no video signal because the set difference result is an empty set (denoted by $\emptyset$).

Since $\tilde{U}^{k+1} \ominus_t \tilde{U}^k$ indicates the missing video contents for those "last-minute" clients started before the transition time t, if these contents are not made up to these clients, disruption will be experienced by them. Fortunately, since this transition means that one channel is to be returned back to the system, we can use this channel to broadcast the missing content before it is returned to the system. For efficiency reason, this channel should be returned as soon as possible. The following definition can further help to expedite this process.

Definition 5: Given $\tilde{U}^{k+1} \ominus_t \tilde{U}^k$, define $PACK(\tilde{U}^{k+1} \ominus_t \tilde{U}^k)$ to be the video stream obtained from the former by removing all its empty contents (i.e., $\emptyset$) and sequentially shifting the remaining non-empty contents to the time points as early as possible after time t.

For instance, Fig. 6(c) shows the result after packing the video stream in Fig. 6(b) up to time t. Intuitively, we try to broadcast the make-up video as early as possible. After all meaningful video data ($\neq \emptyset$) is broadcast, the channel can be released.

Now consider the NCT problem to transit from $\tilde{U}^k$ to $\tilde{U}^{k'}$, $k > k'$. The server should perform the following steps.

1) Let the current time be t. Determine the transition time point t' to be the beginning of the nearest coming time slot in $\tilde{U}^{k'}$.

2) Starting from t', stop broadcasting $\tilde{U}^k$ but instead broadcasting $\tilde{U}^{k'}$.

3) Utilize the $k - k'$ yet-to-be-returned channels, each broadcasting $PACK(\tilde{U}^i \ominus_{t'} \tilde{U}^{i-1}), i = k, \ldots, k'+1$. After the con-

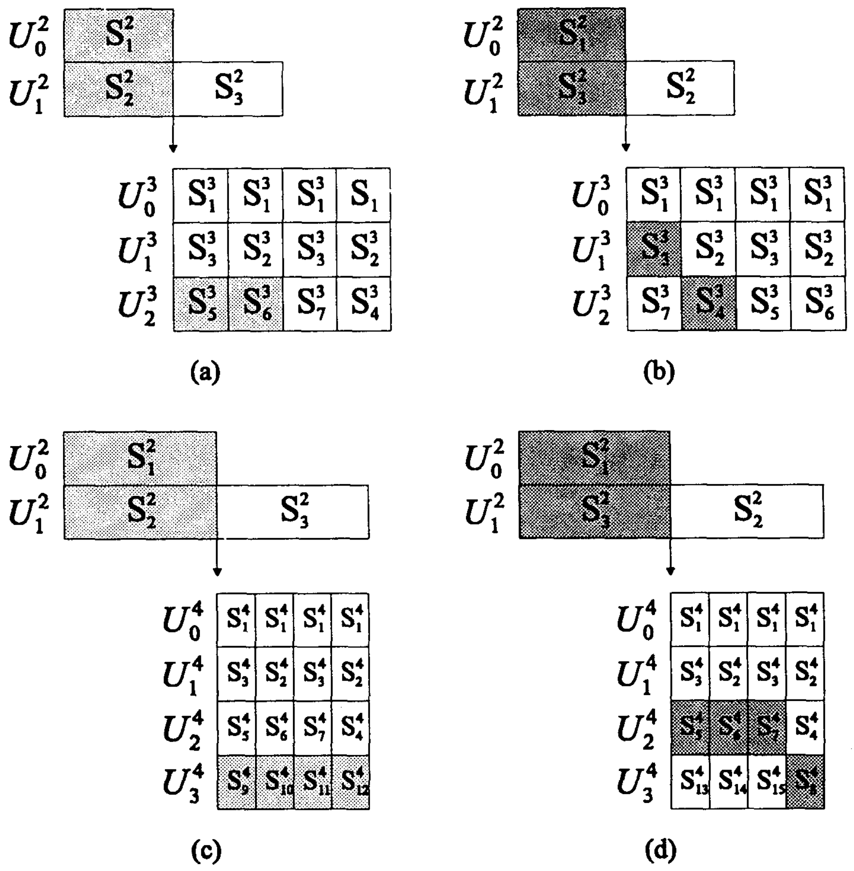

Fig. 5. How a client finding missing segments after PCT transitions: (a)–(b) two possible transitions from $\tilde{U}^2$ to $\tilde{U}^3$; (c)–(d) two possible transitions from $\tilde{U}^2$ to $\tilde{U}^4$. The arrows indicate where the transitions happen.

tent in $PACK(\tilde{U}^i \ominus_{t'} \tilde{U}^{i-1})$ is broadcast, the corresponding channel can be returned back to the system.

For instance, to transit from $\tilde{U}^4$ to $\tilde{U}^2$ at time t, the server should broadcast $\tilde{U}^2$ as well as $PACK(\tilde{U}^4 \ominus_t \tilde{U}^3)$ and $PACK(\tilde{U}^3 \ominus_{t'} \tilde{U}^2)$. The latter two channels can be released after their contents are sent out. An example is illustrated in Fig. 7.

Finally, we comment that our approach does not have the deadlock problem mentioned in Section II. The reason is that we do not need extra channels in addition to those currently used by the video to perform NCT. So it is guaranteed that some channels will be released in finite time. After we obtain some free channels, PCT then can be performed.

IV. Channel Allocation for the FB Scheme

Given K channels, in this section we consider how to assign these channels to m popular videos $V_1, V_2, \dots, V_m$ each of lengths $D_1, D_2, \dots, D_m$ and with request arrival rates $\lambda_1, \lambda_2, \dots, \lambda_m$, respectively. The goal is to minimize the Eq. (1) in Section II. However, since some dummy data is padded to each video, the goal should be slightly changed as follows. Let k_i be the number of channels assigned to $V_i, i = 1..m$. Then its slot time will be δ'_i (as formulated in Eq. (2)). So the goal is to minimize

$$\begin{aligned}&\sum_{i=1}^{m} \lambda_i * \frac{\delta'_i}{2}\\ =& \sum_{i=1}^{m} \lambda_i * \frac{\frac{2^\alpha}{2^\alpha-1} \cdot D_i}{2 \cdot 2^{k_i}} \\ =& \frac{2^\alpha}{2(2^\alpha-1)} \sum_{i=1}^{m} \frac{\lambda_i D_i}{2^{k_i}}\end{aligned} \tag{9}$$

subject to $\sum_{i=1}^{m} k_i = K$ and $k_i \geq \alpha$.

Minimizing Eq. (9) is a non-linear integer programming problem. Below, we propose a greedy approach to this problem. First, each video should be given α channels. Then, we proceed as follows: for each of the remaining unassigned channels, we can calculate the reduction of overall waiting time if this channel is assigned to V_i for $i = 1..m$. Then the channel is assigned to the video which leads to the most reduction. The process is repeated until all channels are assigned.

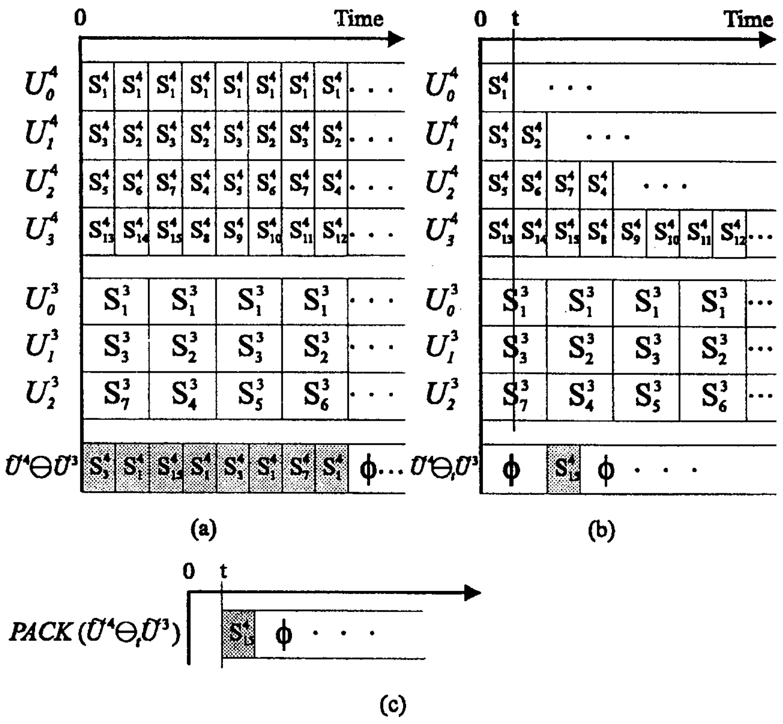
Fig. 6. Examples of (a) $\tilde{U}^4 \ominus \tilde{U}^3$, (b) $\tilde{U}^4 \ominus_t \tilde{U}^3$, and (c) $PACK(\tilde{U}^4 \ominus_t \tilde{U}^3)$.

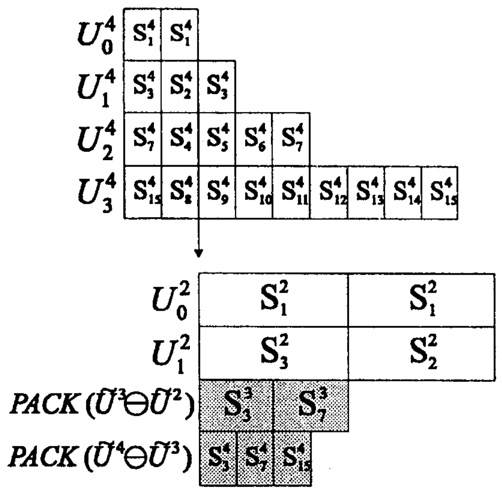
Fig. 7. An example of NCT transiting from $\tilde{U}^4$ to $\tilde{U}^2$.

Theorem 2: The above greedy approach guarantees Eq. (9) to be minimal.

Proof: For each V_i, with α channels, it incurs waiting time of $\lambda_i D_i / 2^\alpha$. Now we list for each V_i the amount of waiting time that can be reduced from α channels to $\alpha + 1$ channels, from $\alpha + 1$ channels to $\alpha + 2$ channels, etc.:

$$V_1 : \frac{\lambda_1 D_1}{2^{\alpha+1}}, \frac{\lambda_1 D_1}{2^{\alpha+2}}, \frac{\lambda_1 D_1}{2^{\alpha+3}}, \cdots$$
$$V_2 : \frac{\lambda_2 D_2}{2^{\alpha+1}}, \frac{\lambda_2 D_2}{2^{\alpha+2}}, \frac{\lambda_2 D_2}{2^{\alpha+3}}, \cdots$$
$$\cdots$$
$$V_m : \frac{\lambda_m D_m}{2^{\alpha+1}}, \frac{\lambda_m D_m}{2^{\alpha+2}}, \frac{\lambda_m D_m}{2^{\alpha+3}}, \cdots$$

Minimizing Eq. (9) is equivalent to maximizing the total amount of reduction. So the problem becomes choosing $K - m\alpha$ numbers from the above sequences, each from the beginning, such that their total is maximal. Since each sequence is descending, it is easy to see that the above proposed greedy approach will achieve this goal. □

The video server should estimate the current request arrival rates of all videos from time to time. Then it should determine a best channel assignment under the current state based on the above greedy method. If changes have to be made, the PCT and NCT can be called. Alternatively, the system manager can manually involve in the assignment (such as increase the α for some particular videos, favor one video over another, etc.).

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 identified an important problem in existing broadcasting schemes — they all assumed that a fixed number of channels/bandwidth will be assigned to a video throughout the broadcasting. If the level of demand on the video changes, either disruption will be experienced by existing clients or extra channels are needed to concurrently serve both existing and new clients during the transition. More seriously, in the latter case, deadlock may even occur, making transition impossible. We have taken the first step and shown how to modify the Fast Broadcasting (FB) scheme to achieve a seamless transition without causing the above problems. We have also looked at this problem from the management level by formulating the channel allocation problem when multiple popular videos are considered. A greedy scheme that can lead to the minimal average viewers' waiting time is given.

Implementations of the proposed schemes are underway at the National Central University. We are also looking at other existing broadcasting schemes regarding the channel transition problem.

REFERENCES

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *IEEE Proceedings of the International Conference on Multimedia Computing and Systems*, pages 118–126, Jun 1996.

[2] L. Atzori, F. D. Natale, M. D. Gregorio, and D. D. Giusto. Multimedia information broadcasting using digital TV channels. *IEEE Transactions on Broadcasting*, 43(3):242–251, Sep 1997.

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

[4] T. Chiueh and C. Lu. A periodic broadcasting approach to video-on-demand service. *SPIE*, 2615:162–169, Oct 1995.

[5] M. Cominetti, V. Mignone, A. Morello, and M. Visintin. The european system for digital multi-programme television by satellite. *IEEE Transactions on Broadcasting*, 41(2):49–62, Jun 1995.

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

[7] W. Hodges, S. Manon, and J. P. P. Jr. Video on demand: Architecture, systems, and applications. *SMPTE Journal*, pages 791–803, 1993.

[8] L.-S. Juhn and L.-M. Tseng. Fast broadcasting for hot video access. In *RTCSA'97*, pages 237–243, Oct 1997.

[9] L.-S. Juhn and L.-M. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Transactions on Broadcasting*, 43(3):268–271, Sep 1997.

[10] L.-S. Juhn and L.-M. Tseng. Enhanced harmonic data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Consumer Electronics*, 44(2):343–346, May 1998.

[11] L.-S. Juhn and L.-M. Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Broadcasting*, 44(1):100–105, Mar 1998.

[12] T. L. Kunii, et al. Issues in storage and retrieval of multimedia data. *Multimedia Systems*, 3(5):298–304, 1995.

[13] T. D. C. Little and D. Venkatesh. Prospects for interactive video-on-demand. *IEEE Multimedia*, 1(3):14–24, March 1994.

[14] B. Ozden, R. Rastogi, and A. Silberschatz. On the design of a low cost video-on-demand storage system. *Multimedia Systems*, 4(1):40–54, 1996.

[15] W. D. Sincoskie. System Architecture for a Large Scale Video On Demand Service. *Computer Networks and ISDN systems*, 22:565–570, 1991.

[16] S. Viswanathan and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *Multimedia Systems*, 4:197–208, 1996.