Data Broadcasting and Seamless Channel Transition for Highly Demanded Videos

Yu-Chee Tseng, Ming-Hour Yang, Chi-Ming Hsieh, Wen-Hwa Liao, and Jang-Ping Sheu, *Senior Member, IEEE*

***Abstract*—One way to broadcast a popular video is to use a number of dedicated channels, each responsible for broadcasting some portion of the video periodically in a predefined way. The stress on channels can be alleviated, and new viewers do not have to wait long to start their playback. Many approaches falling in this category have been proposed. One such scheme that interests us is the fast broadcasting (FB) scheme, which can broadcast a video using k channels by incurring at most $O(D/2^k)$ waiting time on new-coming viewers, where D is the length of the video. In this paper, we consider a set of videos, each being broadcast by the FB scheme. Since the levels of demands on these videos may change by time, it is sometimes inevitable to change the numbers of channels assigned to some videos. We propose a novel *seamless channel transition* enhancement on top of the FB scheme to dynamically change the number of channels assigned to a video on-the-fly. Clients currently viewing this video will not experience any disruption because of the transition. A channel allocation scheme is also proposed based on the arrival rates of videos to minimize the average waiting experienced by all viewers. From the system manager's point of view, the enhancement will make the FB scheme more attractive.**

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

I. Introduction

VIDEO has become one of the most important forms of media in our lives. On average, each household has 1.4 television sets in the U.S. [3]. With the availability of networks, people also wish to be able to access videos instantly at the touch their fingertips. With the advancement of broad-band networking technology, computing technology, and storage technology, VOD (video-on-demand) services have become possible [18], [20]. 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 [4], [11], [23]. The simplest scheme is to dedicate a channel to each client [8], [19]. Many VCR-like functions may be provided (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 input–output (I/O) bandwidth for each video stream before committing to a client's request [10]. Apparently, such systems may easily run out of channels because the growth of the number of channels can never keep up with the growth of the number of clients. This results in tremendous demand for communication bandwidths on the system.

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 when a channel is available [1], [6], [7]. A scheduling policy based on the arrival of the requests to the video is required to best utilize the channels. Two *patching* schemes [12], [9] are proposed on top of the batching approach to allow late-coming clients to join the service with some buffering space and server channel constraints. In the *broadcasting* approach, the server uses multiple dedicated channels to broadcast a video cooperatively. Each channel is responsible for broadcasting some portion of the video. Each client follows some reception rule to grab data from appropriate channels so as to play the whole video continuously. The server's broadcasting activity is independent of the arrivals of requests. Such an approach is more appropriate for popular or hot videos that may interest many viewers at a certain period of time. According to [6] and [7], 80% of demands are on a few (10 or 20) very popular videos.

Because of the efficiency in channel usage, many broadcasting approaches have been proposed [2], [5]–[7], [10], [13]–[17], [21], [22], [24]. The simplest solution is to periodically broadcast the video on several channels, each differentiated by some time [5]. The enhanced batching scheme [6] proposes to divide the video into equal-length segments; a user has to wait no longer than the length of the segment. Many schemes have been proposed by imposing a larger client receiving bandwidth and an extra buffering space at the client side. A scheme called *pyramid* is proposed in [24], which 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 [13], 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 [14], [17]

Paper approved by M. R. Civanlar, the Editor for Image Processing of the IEEE Communications Society. Manuscript received October 21, 1999; revised October 25, 2000. This work was supported by the National Science Council of the Republic of China under Grant NSC89-2213-E-008-023 and Grant NSC89-2213-E-008-025, and the Ministry of Education, R.O.C., under Grant 89-H-FA07-1-4 (Learning Technology).

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

M.-H. Yang, C.-M. Hsieh, W.-H. Liao, and J.-P. Sheu are with the Department of Computer Science and Information Engineering, National Central University, Chung-Li 32054, Taiwan, R.O.C.

Publisher Item Identifier S 0090-6778(01)05003-6.

and the PAGODA scheme [21], [22], 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 [15] and [16]. This scheme is recently proved to be optimal with respect to the bandwidth requirement and the viewers' waiting time [26].

This paper is motivated by the observation that all the above broadcasting schemes [2], [5]–[7], [10], [13]–[17], [21], [22], [24] are aimed at reducing the viewer waiting time given a *fixed* bandwidth for *one* video. If we consider a set of videos supported by a server, the bandwidth received by each video should, to a certain degree, reflect the "level of hotness" of the video. Furthermore, since the hotness of a video will be changed by many factors (such as day of the week, time of day, or social events), the bandwidth assigned to each video may need to be adjusted to reflect the change. It will be certainly desirable for a broadcasting scheme to be able to dynamically adjust the bandwidth assigned to a video on-the-fly in a seamless manner. As far as we know, these issues have not been addressed for any of the above reviewed schemes yet.

In this paper, we pick the FB scheme [14], [17] as our target. We propose a *channel transition* enhancement 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 the clients currently viewing this video will not experience any disruption because of the transition. We note that seamless channel transition is sometimes critical for popular videos, which keep on having new viewers arriving. For instance, let us assume a server that can support as many as seven channels concurrently. Suppose that initially three and four channels are used to broadcast videos V_1 and V_2, respectively, based on the FB scheme. Now, for some reason, suppose that we would like to reconfigure the assignment by using four and three channels for V_1 and V_2, respectively. One naive solution is to simply block all new viewers of V_1 and V_2 and wait for all old viewers to complete. After awhile, all seven channels will be free, and then we can start with the new channel configuration. Unfortunately, the transition time will be roughly half of the length of the longer video, because the longest period to broadcast video segments in the FB scheme is about half of the video length. For instance, for a typical 120-min movie, no new viewers can be accepted for about 60 min, which is intolerable. A less naive solution is to block new viewers of V_2 to vacate the four channels occupied by V_2. After a while, we will have four free channels, by which we can serve new viewers of V_1 with the new configuration. Then we wait further for a while until the original three channels occupied by V_1 are released. Then we can start serving new viewers of V_2. So no new viewers of V_1 will be blocked, but new viewers of V_2 will not be accepted for about half of the total length of V_1 and V_2. This example further justified the importance of our yet-to-be-presented seamless channel transition scheme.

With our enhancement, given some bandwidth and a set of popular videos, the system manager will be able to dynamically and seamlessly change his/her channel assignment policy so as to make the most benefit out of the broadcasting service. We also propose a channel assignment policy, based on the request arrival rates to videos, to minimize the average waiting time incurred on all new viewers. From a system manager's point of view, these enhancements will certainly make the FB scheme more attractive.

Finally, we remark that one reason for us to pick the FB as our target is its simplicity and practicality. Some implementation details and experiences of FB's implementation have been reported recently in [25]. Some discussions on this are in Section II.

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. The proposed seamless channel transition enhancement for the FB scheme is presented in Section III. How to assign channels based on our enhancement is discussed in Section IV. Conclusions are drawn in Section V.

II. Backgrounds and Motivations

A. Review of the FB Scheme

Below, we review the FB scheme proposed by [14] and [17]. 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 $D = 120$ min to be played at rate $b \approx 10$ Mb/s.) 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 $S_1 \circ S_2 \circ \cdots \circ S_N$.

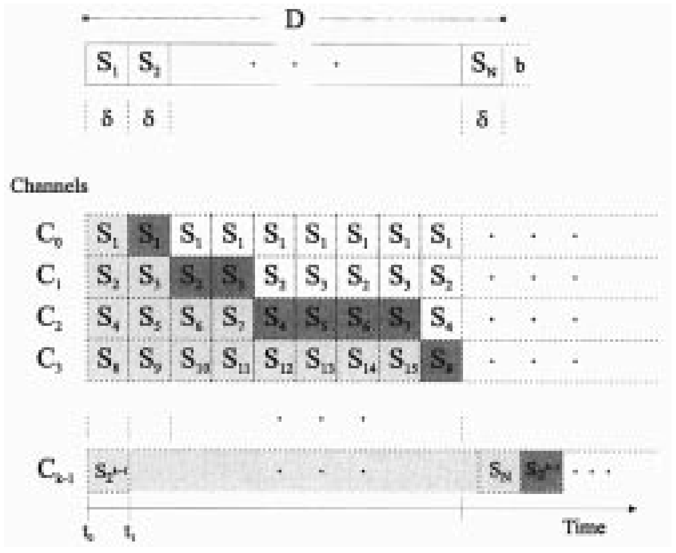


Fig. 1. Data scheduling of the FB scheme.

Let us 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-min video, with five channels, a viewer has to wait no more than $120/(2^5 - 1) < 4$ min to start the service, and with six channels, the maximal waiting time further reduces to $120/(2^6 - 1) < 2$ min.

What the client has to pay for are a larger receiving bandwidth and an extra buffer space for those premature segments. The receiving bandwidth is the number of channels for the video. The buffer space will depend on the time slot when a client begins its service. The maximum buffer requirement is less than half of the video size, $Db/2$[14]. 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.

Implementation experiences of the FB scheme have been reported in [25]. The MBone (VIC H.261) protocol is used to transmit video packets to clients. The prototype is called *Listener*. It is built from pure Microsoft Win32 native codes with the Microsoft Foundation Class Library. Listener is composed of three independent Win32 applications: RTP media emitter, RTP packet recorder, and RTP media player. The RTP media emitter fetches prerecorded RTP packets from its hard disk driver and injects them into the network. The RTP packet recorder receives video/audio packets and places them in its disk through the WIN98 FAT32 file system. The RTP media player is then responsible of the playback. The system runs on a 10-Mb/s Ethernet supported by TCP/IP (some transformation is applied to simulate a multichannel environment by the shared Ethernet). The detailed hardware and software descriptions of the system are shown in Table I. Under such environment, around 16 channels can be support by the video server. For example, it is shown that the server can concurrently support three RTP emitters (i.e., three videos) each with five channels, or four RTP emitters (i.e., four videos) each with four channels smoothly. A client can receive segments from 16 channels concurrently without problem. So the current bottleneck is on the server's side to support more channels.

B. Problems with the FB Scheme

This section motivates the work of this paper. We observe that the FB scheme only considers the scheduling of segments of one video on a given set of channels. The number of channels assigned to the video is fixed. However, since the level of demand on the video may change by time, it is desirable that the number of channels assigned to the video can adapt to such changes. We call this *channel transition* when the video is undergoing some change on the number of channels used.

One trivial solution to channel transition is to allocate a new set of channels for the new configuration to serve new-coming viewers, while keeping the old set of channels unchanged to serve the old viewers currently receiving the video segments. Since we assume that the video is very popular, it is likely that we have viewers starting their playback at the beginning of every time slot. So in the worst case, the old set of channels may be released after $2^{k-1}\delta$ time (which is the period of the channel C_{k-1}). This is larger than half of the length of the video. Thus,

TABLE I
HARDWARE AND SOFTWARE DESCRIPTIONS ON WHICH THE PROTOTYPE LISTER RUNS

Items	Description
Operating system	Microsoft Windows 98 SE
CPU	Intel Celcern 800 MHz
Hard disk driver	Generic IDE H.D. (7200 rpm)
Network	A separated IEEE 802.3 10 Mbps Ethernet
Audio	ADPCM (length 1602 sec, average bit rate 32 Kbps)
Video	H. 261 video (length: 1602 sec, average bit rate 514 kbps)

the waste of communication bandwidth will be significant, especially when the video is long.

More seriously, the above naive transition is only possible if we have spare free channels for the new configuration. If we consider a set of videos supported by a server, there may not always exist sufficient spare channels for the transition. If so, we may have to block the new viewers of some videos so as to vacate their channels for the transition. Again, the blocking time could be longer than half of the length of the video, or even much longer (refer to the example in Section I). This amount of waiting time will be intolerable if the video is long (e.g., for a typical 120-min video, the waiting time will be longer than an hour).

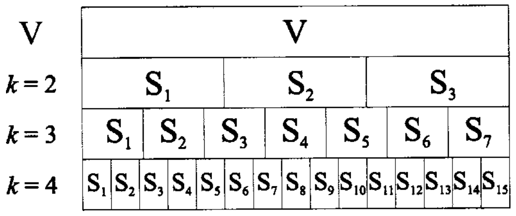
Therefore, it is desirable to have a scheme to dynamically adjust the number of channels assigned to a video on-the-fly. The transition should be *seamless*, in the sense that during the transition period, clients currently viewing the video will not experience any disruption. Meanwhile, new viewers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

III. CHANNEL TRANSITION FOR THE FB SCHEME

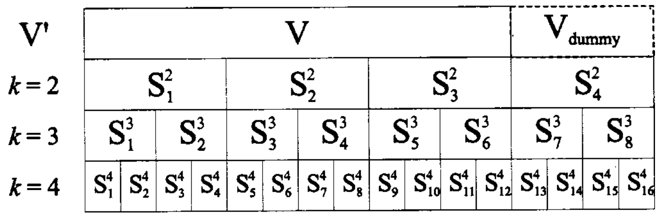
In this section, we will show how to modify the FB scheme to make seamless channel transition possible. We consider one video V of length D. First, V will be padded with some dummy video stream at its end before being broadcast (Section III-A). Then, we will establish the relationship of channel contents 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, three and four 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 among the segments obtained from these two channel assignments. This is similar for most other assignments. Below, to establish some correspondence relationship among segments of different channel assignments, we will pad some dummy data at the end of V. By "correspondence relationship," we mean that a segment of one channel assignment can be represented by multiple segments of another channel assignment. For example, in Fig. 2, we show the relationships of segments in the original FB scheme and after our data padding. The segments with data padding are aligned in a better way (we will discuss this in a more formal way later).



(a)


V'
V
V dummy
k = 2
S^2_1 S^2_2 S^2_3 S^2_4
k = 3
S^3_1 S^3_2 S^3_3 S^3_4 S^3_5 S^3_6 S^3_7 S^3_8
k = 4
S^4_1 S^4_2 S^4_3 S^4_4 S^4_5 S^4_6 S^4_7 S^4_8 S^4_9 S^4_10 S^4_11 S^4_12 S^4_13 S^4_14 S^4_15 S^4_16

(b)

Fig. 2. The relationship of segments when k $(= 2, 3, 4)$ channels are assigned to a video V: (a) in the original FB scheme and (b) after our data padding. It is assumed that $\alpha = 2$, so $|V_{\text{dummy}}| = D/3$.

Now, assume that α is a fixed small integer, which indicates the minimum number of channels that must be assigned to V. Since our basic assumption is that V is a highly demanded video, it is reasonable to assume that α is at least 3 or 4 (however, note that our scheme will work for any fixed α). For instance, for a typical 120-min video, $\alpha = 3$ gives an average waiting time of $120/(2 \cdot (2^3 - 1)) \approx 8.57$ min, and $\alpha = 4$ an average waiting time of $120/(2 \cdot (2^4 - 1)) = 4$ min. At the end of V, a dummy video stream V_{dummy} of length

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

is padded. Now let

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

Suppose that k channels $C_0, C_1, \ldots, C_{k-1}$ are assigned to V, where $k \geq \alpha$. 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., $S_1 \circ S_2 \circ \cdots \circ S_N = V'$.

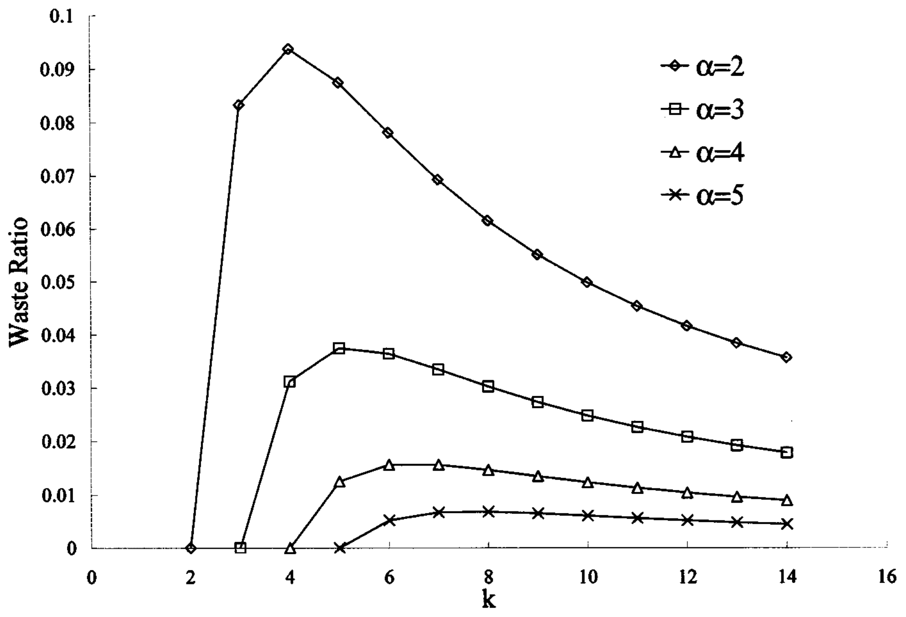


Fig. 3. The ratio of bandwidth waste due to data padding for a 120-min video using k channels.

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

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

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. One can easily verify the correctness of the above modification.

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 us 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. 2(b). 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. The correspondence relation is formally derived in the following lemma.

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

Below, we comment on the extra cost of the data padding. First, we consider the waste of communication bandwidth occupied by these dummy segments. We are using k channels to broadcast V'. The last $2^{k-\alpha} - 1$ segments contain dummy videos. Note that these dummy segments must be broadcast on the last channel C_{k-1}. Taking one period of the last channel C_{k-1} as the denominator, we can see that the following ratio of bandwidths of the k channels are used to deliver dummy video segments

$$\frac{2^{k-\alpha} - 1}{k \cdot 2^{k-1}}.$$

In Fig. 3, we show the ratio of bandwidth waste for different k and α. The highest ratio is around 9%, and the ratio decreases fast as k enlarges. A larger α will incur less waste. We remark that one possible application of these dummy segments is to deliver advertisements or previews of coming-up videos to the viewers. This is reasonable because we seldom see any broadcasting programs without advertisements.

Second, we evaluate the increase of segment length (and thus viewer waiting time) due to our data padding. 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

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

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-min video, the average waiting

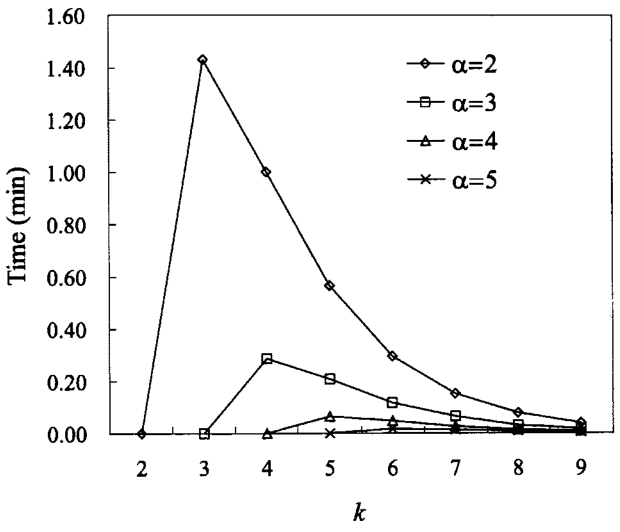


Fig. 4. The increase in average waiting time after data padding for a 120-min video using k channels.

time is increased by 1.43 min when $k = 3$, by 1.00 min when $k = 4$, and by 0.56 min when $k = 5$. This increase is insignificant. If α is larger, the increase is much less. In Fig. 4, we show the increase of the average waiting time at various values of α and k for a 120-min video.

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 $\Psi(t_1, t_2)$.

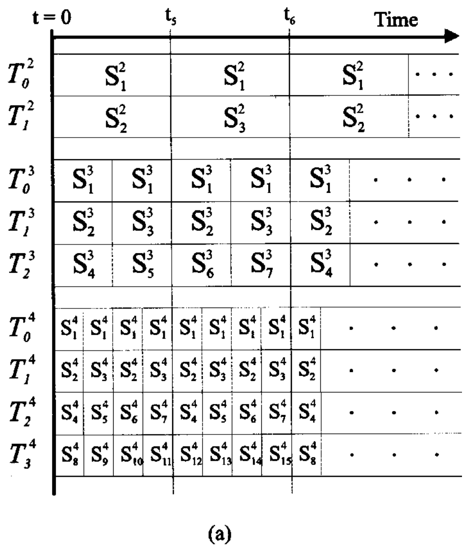
Definition 1: Given an integer k and an integer d such that $k \geq \alpha$ and $0 \leq d < k$, we define the infinite video stream

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

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.



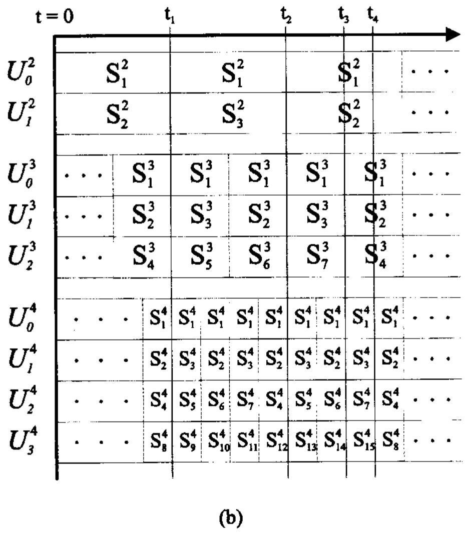


Fig. 5. (a) Channel contents $\tilde{T}^k$ at $k = 2, 3, 4$. (b) Shifted channel contents $\tilde{U}^k$ at $k = 2, 3, 4$. For clarity, the first few segments of $\tilde{U}^k$ are not shown so as to observe the shifted effect.

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

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

where shift$(i, s_1 \circ s_2 \circ \cdots)$ 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. 5 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 us compare Fig. 5(a) and (b). Before the shifting, the *starting time* of the first segment in each channel (i.e., $\{S_1^2, S_2^2\}$, $\{S_1^3, S_2^3, S_4^3\}$, 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 us denote by $U_d^k[j]$ the video content of U_D^k at the jth 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

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

Theorem 1: For any j, we have

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

Intuitively, the above theorem states that the content of U_d^k at the jth 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 jth 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

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

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. 5(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. 5(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)*. The fundamental theory of PCT and NCT is from the above two corollaries. Intuitively, these two corollaries describe why viewers will not experience disruption at PCT and what needs to be made up to viewers at NCT.

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. 6(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. 6(c) and (d).

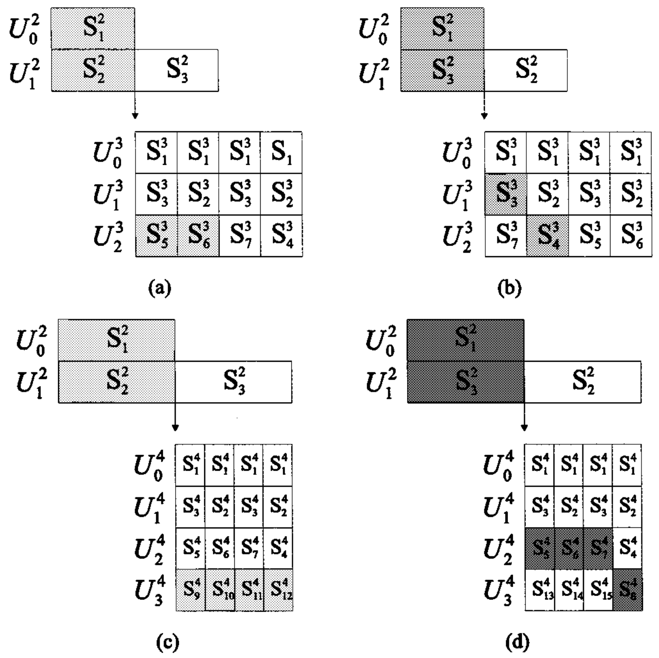


Fig. 6. How a client finding missing segments after PCT transitions: (a)–(b) two possible transitions from $\tilde{U}^2$ to $\tilde{U}^3$; (c)–(d) two possible transitions from $\tilde{U}^2$ to $\tilde{U}^4$. The arrows indicate where the transitions happen.

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 $\setminus$ means *set difference*)

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

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. 7(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

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

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 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 $d = 0 \ldots k$. 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 (2) for clarity). Fig. 7(b) shows an example, which is obtained from

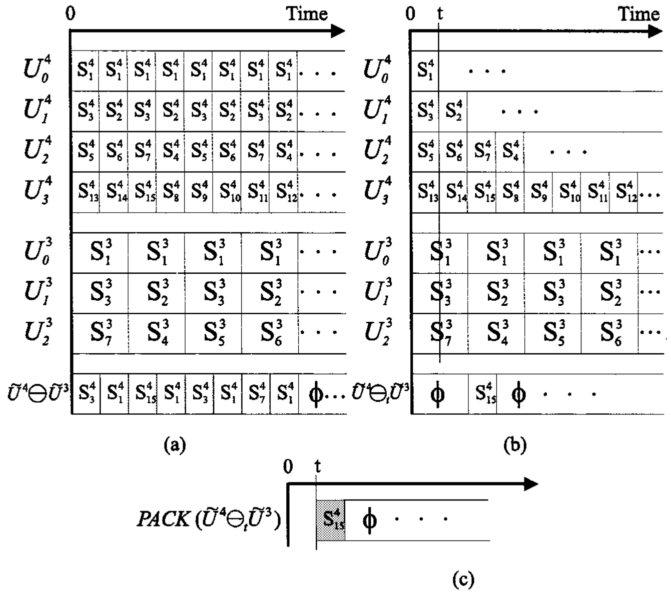


Fig. 7. Examples of (a) $\tilde{U}^4 \ominus \tilde{U}^3$, (b) $\tilde{U}^4 \ominus_t \tilde{U}^3$, and (c) $PACK(\tilde{U}^4 \ominus_t \tilde{U}^3)$.

Fig. 7(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 reasons, 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 nonempty contents to the time points as early as possible after time t.

For instance, Fig. 7(c) shows the result after packing the video stream in Fig. 7(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 content 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

TABLE II
COMPARISON OF BUFFERING SPACES REQUIRED BY THE ORIGINAL FB SCHEME AND OUR NCT SCHEME (IN TERMS OF THE NNUMBER OF MINUTES OF A 120-MIN VIDEO). FB(i) MEANS USING i CHANNELS, AND NCT(i) MEANS TRANSITING FROM $i+1$ CHANNELS TO i CHANNELS

i	2	3	4	5	6	7	8	9
FB(i)	40.00	51.42	56.00	58.06	59.04	59.52	59.76	59.88
Ours(i), $\alpha = 2$	60.00	70.00	75.00	77.5	78.75	79.37	79.68	79.84
Ours(i), $\alpha = 3$		60.67	65.32	67.69	68.82	69.41	69.70	69.85
Ours(i), $\alpha = 4$			60.48	62.73	63.86	64.43	64.71	64.85
Ours(i), $\alpha = 5$				60.27	61.38	61.94	62.32	62.36

after their contents are sent out. An example is illustrated in Fig. 8.

Finally, we comment that our approach does not need to allocate a whole set of spare free channels to perform channel transition. From a video server's point of view, when the videos supported by the server must undergo some channel rearrangement, it can always perform NCT first for those videos whose channels need to be cut. Once sufficient channels are released, it can perform PCT.

3) Extra Costs of NCT: In this section, we discuss some hidden costs of NCT. First, we evaluate the potential increase of buffering space required at the viewer's side due to negative channel transition. We do not have a close formula for this. So we wrote a simulator to broadcast a 120-min movie with $\alpha = 2, 3, 4, 5$. We exhaustedly searched all possibilities to find out the maximum buffering space required at different numbers of channels. The comparison to the original FB scheme is in Table II. About 25%–50% increase of buffering space on top of that required by the FB scheme is required by NCT at $\alpha = 2$. With larger α, the buffering space will converge to that required by the original FB scheme.

Also, when performing NCT, the channels to be released will not be vacated immediately. There will be some delay (to broadcast the content defined by the $PACK()$ function). Again, we used a simulator to broadcast a 120-min movie with $\alpha = 2, 3, 4, 5$. We exhaustedly searched all possibilities to find out the average time required to release the channel when we perform NCT to transit from $i+1$ channels to i channels. The result is shown in Fig. 9.

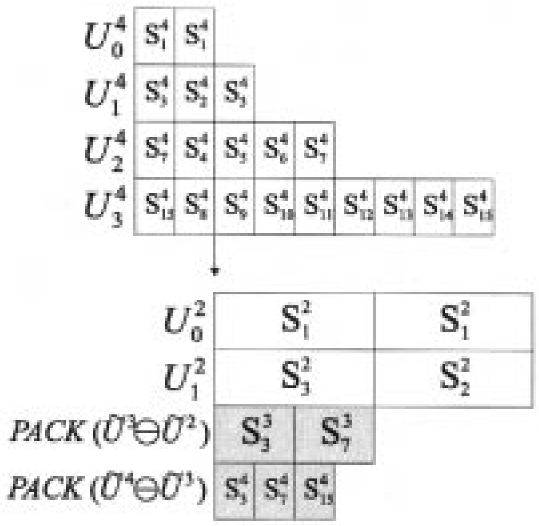


Fig. 8. An example of NCT transiting from $\tilde{U}^4$ to $\tilde{U}^2$.

IV. CHANNEL ALLOCATION POLICY

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. Since the main focus of the FB scheme is new-coming viewers' waiting time, one possibility is to formulate the channel allocation problem based on the average waiting time incurred on all viewers 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 $\delta_i/2 = D_i/(2\cdot(2^{k_i}-1))$. Considering all videos, our goal is to minimize the average waiting time of all viewers, i.e.,

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{m} \lambda_i * \frac{D_i}{2\cdot(2^{k_i}-1)} \\ \text{subject to} \quad & \sum_{i=1}^{m} k_i \le K, \qquad \text{such that } k_i \ge \alpha \end{aligned} \tag{3}$$

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 $\alpha \ge 3$ or 4 (but our formulation will work for any fixed α). For instance, for a typical 120-min video, $\alpha = 3$ gives an average waiting time of $120/(2\cdot(2^3-1)) \approx 8.57$ min, and $\alpha = 4$ an average waiting time of $120/(2\cdot(2^4-1)) = 4$ min. 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.

Next, consider our enhancement on the FB scheme. Given K channels, we still need to assign these channels to the m videos $V_1, V_2, \ldots, V_m$. However, since some dummy data is padded to each video, the goal should be slightly changed as follows. Let D_i' be the length of V_i after padding [refer to (1)]. Let k_i be

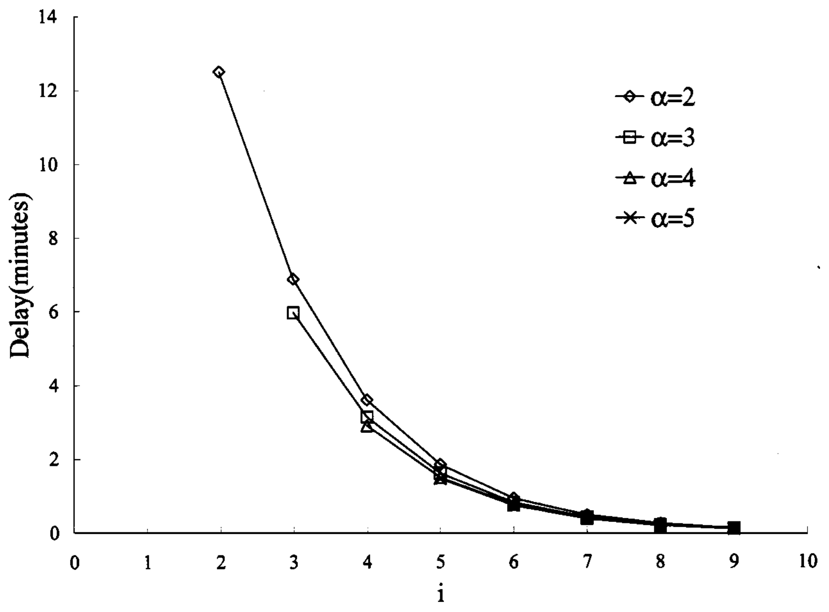


Fig. 9. The average delay to vacate a channel when performing NCT to transit from $i+1$ channels to i channels.

the number of channels assigned to $V_i, i = 1..m$. Then its slot time will be $\delta'_i = D'_i/2^{k_i}$. So the goal should be changed to minimizing

$$\begin{aligned}\sum_{i=1}^{m} \lambda_i * \frac{\delta'_i}{2} = \sum_{i=1}^{m} \lambda_i * \frac{D'_i}{2\cdot 2^{k_i}} = \sum_{i=1}^{m} \lambda_i * \frac{(2^\alpha/2^\alpha - 1)\cdot D_i}{2\cdot 2^{k_i}} \\ = \frac{2^\alpha}{2(2^\alpha - 1)} \sum_{i=1}^{m} \frac{\lambda_i D_i}{2^{k_i}} \qquad (4)\end{aligned}$$

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

Minimizing (4) is a nonlinear 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.

Theorem 2: The above greedy approach guarantees (4) to be minimal.

The video server should estimate the current request arrival rates of all videos from time to time. Then it can determine the best channel assignment policy under the current state using 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.).

Finally, we remark that the above optimization is purely based on viewers' average waiting time. Other optimization goals could exist too (such as imposing some buffering space constraints on viewers, setting an upper bound on the number of channels a video can use, assuming variable viewer arrival rates for videos, and transferring a viewer's waiting time into a more complicated cost function). This could lead to different channel allocation policy and deserves further study.

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 service. If the level of demand on the video changes, the channel allocation has to undergo some transition. Unfortunately, as we have observed, if a naive solution is adopted, either intolerably long waiting time will be incurred on new-coming viewers, or extra spare channels must be reserved in advance to concurrently serve both existing and new-coming viewers during the transition, which will be very costly. We have taken the first step and shown how to modify the FB scheme to achieve a seamless channel transition without causing the above problems. We have also looked at this problem from the system manager's level by formulating the channel allocation policy in terms of viewers' average waiting time when multiple popular videos are considered. Future research can be directed to solving the seamless channel transition problem for the many other broadcasting schemes as reviewed in Section I. Another direction, as suggested by one referee, is to evaluate the waiting time incurred on users based on the popularity of videos, such as the Zipf-like distribution.

Appendix
Proofs of Lemmas and Theorems

Proof of Lemma 2: We show how the subscript of S is derived. The "$2^{k-\alpha} - 1$" is for the shifting effect. The "$\bmod 2^d$" 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 $S_{2^d}^k, S_{2^d+1}^k, S_{2^d+2}^k, \ldots$). □

Proof of Theorem 1: By Lemma 1 and Lemma 2, we have

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

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 (5) to

$$U_d^k[j] = S^{k+1}_{2^{d+1}-1} \circ S^{k+1}_{2^{d+1}}. \tag{6}$$

By Lemma 2, we can derive that

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

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 (5) that

$$\begin{aligned}U_d^k[j] &= S^{k+1}_{((2j-2^{k+1-\alpha}+1)\bmod 2^{d+1})+2^{d+1}+1-1}\\ &\qquad \circ S^{k+1}_{((2j+1-2^{k+1-\alpha}+1)\bmod 2^{d+1})+2^{d+1}}.\end{aligned} \tag{7}$$

The parameter on the left-hand side of $\circ$ is so written because the "$\bmod$" operation will result in a nonzero value, making possible to move a value of 1 outside the "$\bmod$" 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. □

Proof of Theorem 2: 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.,

$$\begin{aligned}V_1 &: \frac{\lambda_1 D_1}{2^{\alpha+1}}, \frac{\lambda_1 D_1}{2^{\alpha+2}}, \frac{\lambda_1 D_1}{2^{\alpha+3}}, \ldots\\ V_2 &: \frac{\lambda_2 D_2}{2^{\alpha+1}}, \frac{\lambda_2 D_2}{2^{\alpha+2}}, \frac{\lambda_2 D_2}{2^{\alpha+3}}, \ldots\\ &\ldots\\ V_m &: \frac{\lambda_m D_m}{2^{\alpha+1}}, \frac{\lambda_m D_m}{2^{\alpha+2}}, \frac{\lambda_m D_m}{2^{\alpha+3}}, \ldots .\end{aligned}$$

Minimizing (4) 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 proposed greedy approach will achieve this goal. □

REFERENCES

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

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

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

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

[5] T. Chiueh and C. Lu, "A periodic broadcasting approach to video-on-demand service," *Int. Soc. Opt. Eng.*, vol. 2615, pp. 162–169, Oct. 1995.

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

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

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

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

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

[11] W. Hodges and S. Manon, "Video on demand: Architecture, systems, and applications," in *Building an Infrastructure for Managing Compressed Video Systems*, 1993, pp. 791–803.

[12] K. Hua, Y. Cai, and S. sheu, "Patching: A multicast technique for true video-on-demand service," in *Proc. ACM Multimedia*, Sept. 1998, pp. 191–200.

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

[14] L.-S. Juhn and L.-M. Tseng, "Fast broadcasting for hot video access," *Real-Time Computing Syst. Applicat.*, pp. 237–243, Oct 1997.

[15] ——, "Harmonic broadcasting for video-on-demand service," *IEEE Trans. Broadcast.*, vol. 43, pp. 268–271, Sept. 1997.

[16] ——, "Enhanced harmonic data broadcasting and receiving scheme for popular video service," *IEEE Trans. Consumer Electron.*, vol. 44, pp. 343–346, May 1998.

[17] ——, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Trans. Broadcast.*, vol. 44, pp. 100–105, Mar 1998.

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

[19] T. D. C. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, vol. 1, pp. 14–24, March 1994.

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

[21] J.-F. Paris, "A simple low-bandwidth broadcasting protocol," in *Proc. Int. Conf. Computer Communication and Network*, 1999, pp. 118–123.

[22] J.-F. Paris, S.-W. Carter, and D.-D. Long, "A hybrid broadcasting protocol for video on demand," in *Multimedia Computing Networking*, 1999, pp. 317–326.

[23] W. D. Sincoskie, "System Architecture for a Large Scale Video On Demand Service," *Comput. Networks ISDN Syst.*, vol. 22, pp. 565–570, 1991.

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

[25] Z.-Y. Yang, "The Telepresentation System over Internet with Latecomers Support," Ph.D. dissertation, Nat'l. Central Univ., Taiwan, R.O.,C., 2000.

[26] Z.-Y. Yang, L.-S. Juhn, and L.-M. Tseng, "On optimal broadcasting scheme for popular video service," *IEEE Trans. Broadcast.*, vol. 45, pp. 318–322, 1999.

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 obtained the Ph.D. degree in computer and information science from the Ohio State University, Columbus, in January 1994.

He worked for D-LINK Inc. as an Engineer in 1990. 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 been a Professor since 1999. Since Aug. 2000, he has been a Professor at the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan, R.O.C. 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 and the Association for Computing Machinery. He served as a Program Committee Member for the 1996 International Conference on Parallel and Distributed Systems, the 1998 International Conference on Parallel Processing, the 2000 International Conference on Distributed Computing Systems, and the 2000 International Conference on Computer Communications and Networks. He was a Workshop Co-Chair of the 1999 National Computer Symposium.

Ming-Hour Yang received the B.S. degree in computer science from Soochow University, Taiwan, R.O.C., in 1994, and the M.S. degree in electronic engineering from the Chung-Hua University, Taiwan, R.O.C., in 1996. He is currently working toward the Ph.D. degree at the Department of Computer Science and Information Engineering, National Central University.

His interests include video on demand, parallel and distributed computing, fault tolerance, mobile computing, and wireless networks.

Chi-Ming Hsieh was born in Taiwan, R.O.C. He reveived the B.S. degree in mathematics and the M.S. degree in computer science and information engineering from National Central University in 1997 and 1999, respectively.

Since October 1999, he has been an Assistant Researcher in the Chung-Shan Institute of Science and Technology, Taiwan, R.O.C. His current research interests include wireless communication and object-oriented software engineering.

Wen-Hwa Liao received the B.S. degree in computer science from Soochow University, Taiwan, R.O.C., in 1989, and the M.S. degree in computer science and information engineering from National Central University, Taiwan, R.O.C., in 1992. He is currently working toward the Ph.D. degree in computer science and information engineering at National Central University, Taiwan.

His research interests include wireless network protocol design, mobile computing, and parallel and distributed computing.

Jang-Ping Sheu (S'85–M'86–SM'98) received the B.S. degree in computer science from Tamkang University, Taiwan, R.O.C., in 1981, and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University, Taiwan, R.O.C., in 1983 and 1987, respectively.

He joined the faculty of the Department of Electrical Engineering, National Central University, Taiwan, R.O.C., as an Associate Professor in 1987. He is currently a Professor with the Department of Computer Science and Information Engineering, National Central University. From July of 1999 to April of 2000, he was a Visiting Scholar at the Department of Electrical and Computer Engineering, University of California, Irvine. His current research interests include parallel proceeing and mobile computing.

Dr. Sheu is a a member of the ACM and Phi Tau Phi Society. He is an Associate Editor of *Journal of Information Science and Engineering* and *Journal of the Chinese Institute of Engineers*. He received the Distinguished Research Awards of the National Science Council of the Republic of China in 1993–1994, 1995–1996, and 1997–1998.