# Seamless Channel Transition for the Staircase Video Broadcasting Scheme

Yu-Chee Tseng, *Senior Member, IEEE*, Yu-Chi Chueh, and Jang-Ping Sheu, *Senior Member, IEEE*

***Abstract*—In the literature, many broadcasting-based schemes have been proposed to efficiently support near-VOD services. However, none of these schemes allows the server to dynamically and seamlessly change the number of channels allocated to a video. Naively allocating a new set of channels for the transition could increase server's load, waste communication bandwidth, and even drain the channels of the system. In Tseng *et al.* (2000), it is shown how to enhance the Fast Broadcasting (FB) scheme for seamless channel transition. The problem remains open whether other broadcasting-based schemes can sustain seamless channel transition. In this paper, we show how to enhance the Staircase Broadcasting (SB) scheme so that a server can seamlessly increase or decrease the channels allocated to a video. The SB scheme has been proved to require significantly less buffering space than FB, while sustaining the same startup latency as FB. Detailed performance comparisons are presented to demonstrate the advantages of the proposed scheme.**

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

## I. Introduction

THE growth of broadband networks has made multimedia services possible. Among the possible broadband services, video-on-demand (VOD) is perhaps one of the most demanding, but challenging, one [5] [24]. VOD services would allow users to on-line access videos from a video library and can support interactive TV. Since this is very resource-demanding on the video servers and the network bandwidth, intensive research has been devoted to related issues recently.

A VOD system is typically implemented by a client-server architecture supported by certain transport networks. This paper focuses on the network and transportation parts. Depending on how communication channels are used, a VOD system can be classified as *interactive* or *broadcasting*. An interactive approach [10], [20] serves each client (or a group of clients arriving close in time) with a dedicated channel. So many VCR functions (forward, rewind, pause, search, etc.) can be emulated. However, 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.

To relieve the stress on channels and I/O demands, the broadcast-based approach [3], [7] delivers video contents independent of clients' requests. This approach is more appropriate for popular (or hot) videos that may interest many viewers at a certain period of time. Many approaches fall into this category [2], [6], [8], [9], [11], [14]–[16], [18], [19], [22], [23], [26]. A simple solution is to periodically broadcast the video on several channels, each differentiated by some time [6]. The *batching* approach collects a group of requests that arrive close in time, and serves them all together when a channel is available [1], [8], [9]. A scheduling policy considering the arrival of requests is required to best utilize the channels. Two *patching* schemes [12], [13] are proposed to allow late-coming clients to join the service with some buffering space and server channel constraints.

One promising direction to support highly demanded videos is to partition a video into multiple segments and arrange them for broadcasting in a certain manner. The EB scheme [8] proposes to divide the video into equal-length segments; a user has to wait no longer than the length of one segment. Many schemes have been proposed by imposing a larger client receiving bandwidth and an extra buffering space at the client side. A scheme called *Pyramid* is proposed in [26], 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 [14], and *greedy disk-conserving* scheme [11] to address the disk buffering requirement at the client side. A number of works have dedicated to reducing the waiting time experienced by viewers. Two instances are the Fast Broadcasting (FB) scheme [15], [19] and the PAGODA scheme [22], [23], which can broadcast a video using $k$ channels by having new-coming viewers to wait no longer than $\Theta(D/2^k)$ and $\Theta(D/5^{k/2})$ time, respectively, where $D$ is the length of the video. A *harmonic* scheme based on the concept of harmonic series is proposed in [16], [18]. This scheme is recently proved to be optimal with respect to the bandwidth requirement and the viewers' waiting time [27]. Broadcasting may suffer from the packet loss problem, and the reliability issue is addressed in [21]. Stream merging algorithms, which also intend to save the server's traffic load, are discussed in [4]. Such schemes try to merge different user groups lagged by time. However, since each new client may occupy a dedicate channel, the approach is not appropriate to support videos that are very popular.

This paper is motivated by the observation that all the above broadcasting schemes [2], [6], [8], [9], [11], [14]–[16], [18], [19], [22], [23], [26] aim at reducing the waiting time or buffering space incurred on clients given a fixed bandwidth for

Manuscript received January 18, 2002; revised November 18, 2002 and July 24, 2003; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. Ross. This work was supported by the MOE Program for Promoting Academic Excellence of Universities under Grants A-91-H-FA07-1-4 and 89-E-FA04-1-4.

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

Y.-C. Chueh and J.-P. Sheu are with the Department of Computer Science and Information Engineering, National Central University, Chung-Li 32054, Taiwan, R.O.C. (e-mail: sheujp@csie.ncu.edu.tw).

Digital Object Identifier 10.1109/TNET.2004.828965

TABLE I
COMPARISON OF BROADCASTING SCHEMES. ($D$ = VIDEO LENGTH; $k$ = NUMBER OF CHANNELS)

| scheme | waiting time | buffer req. | seamless transition |
|---|---|---|---|
| FB [19] | $D/(2^k-1)$ | $(2^{k-1}-1)D/(2^k-1)$ | no |
| SB [17] | $D/(2^k-1)$ | $(2^{k-1}-1)D/(2^{k+1}-2)$ | no |
| seamless FB [25] | $(D \times \frac{2^\alpha}{(2^\alpha-1)})/2^k$ | $\frac{2^{k-1}-1}{2^k} \times \frac{2^\alpha}{2^\alpha-1} \times D$ | yes |
| seamless SB | $D/(3 \times 2^{k-2})$ | $D/4 + D/(3 \times 2^{k-1})$ | yes |

*one* video. If we consider *multiple* videos supported by a server, the bandwidth allocated to 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 a week, time of a day, and social events), the bandwidth assigned to each video may need to be adjusted to reflect the change. It is certainly desirable for a broadcasting scheme to be able to dynamically adjust the bandwidth assigned to each video *on-the-fly* in a seamless manner. The is achieved by the seamless FB (SFB) scheme [25], which can seamlessly transit the number of channels assigned to a video for the FB scheme such that the clients currently viewing this video (based on the FB scheme) will not experience any disruption because of the transition. However, it remains open whether it is possible to enhance other broadcasting schemes with such seamless transition property.

In this paper, we show how to achieve seamless channel transition for the Staircase Broadcasting (SB) scheme [17]. It is worth noting that the SB scheme is an extension of the FB scheme [15], [19]. Both SB and FB have the same waiting time, but SB significantly improves over FB in the buffering space requirement. It is thus desirable to enhance SB for seamless transition from the system manager's point of view. Our new scheme, called seamless SB (SSB) scheme, slightly modifies the arrangement of segments in the SB scheme to achieve this goal. In Table I, we compare the FB, SB, SFB, and SSB schemes. The main contribution of SSB is an increase in waiting time (approximately by a factor of $1/3$) as opposed to SB in trade of the seamless transition property. SSB also maintains the small buffering requirement of the original SB scheme. Both SB and SSB need to buffer about $1/4$ of the video as opposed to FB and FFB, which require to buffer about $1/2$ of the video. More detailed performance comparison is in Section IV. SFB and SSB have the similar structures (consisting of positive channel transition and negative channel transition parts). However, the segment arrangements in these two schemes are very different.

The rest of this paper is organized as follows. Section II reviews the FB and SB schemes. The proposed SSB scheme is presented in Section III. Performance comparisons are given in Section IV. A channel allocation policy considering multiple videos is presented in Section V. Conclusions are drawn in Section VI.

## II. REVIEWS

In this section, we review the FB and the SB schemes, which are essential to establish our seamless channel transition result.

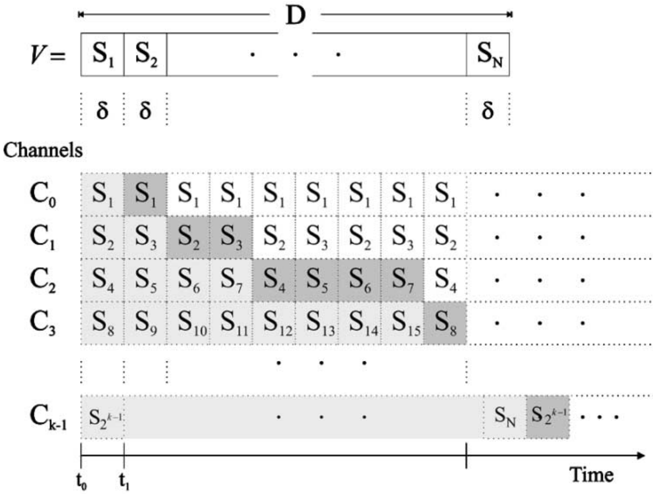

D
$V$=
$S_1$ $S_2$ $S_N$
δ δ δ
Channels
$C_0$ $S_1$ $S_1$ $S_1$ $S_1$ $S_1$ $S_1$ $S_1$ $S_1$ $S_1$
$C_1$ $S_2$ $S_3$ $S_2$ $S_3$ $S_2$ $S_3$ $S_2$ $S_3$ $S_2$
$C_2$ $S_4$ $S_5$ $S_6$ $S_7$ $S_4$ $S_5$ $S_6$ $S_7$ $S_4$
$C_3$ $S_8$ $S_9$ $S_{10}$ $S_{11}$ $S_{12}$ $S_{13}$ $S_{14}$ $S_{15}$ $S_8$
$C_{k-1}$ $S_{2^{k-1}}$ $S_N$ $S_{2^{k-1}}$
$t_0$ $t_1$
Time

Fig. 1. Video partitioning and scheduling of the FB scheme.

The FB scheme partitions the video only vertically, while the SB scheme does so both vertically and horizontally.

### A. Fast Broadcasting (FB) Scheme

The FB scheme is designed with both efficiency and simplicity in mind. We are given a video $V$ of length $D$. Since it is assumed that $V$ is very popular, providing each client a dedicated channel to view $V$ is infeasible. To resolve this problem, FB assumes only a fixed number of $k$ channels, $C_0, C_1, \ldots, C_{k-1}$, to support $V$. Each channel has sufficient bandwidth to broadcast $V$. However, instead to letting each channel broadcast $V$ completely, these channels will cooperate in a certain manner. Specifically, the video server uses the following rules to broadcast $V$.

1) Partition $V$ evenly into $N$ fixed-length 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 $\delta$. 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. Note that none of these channels broadcasts the complete video $V$.

To view $V$, a client should monitor and receive segments from all $k$ channels according to the following rules.

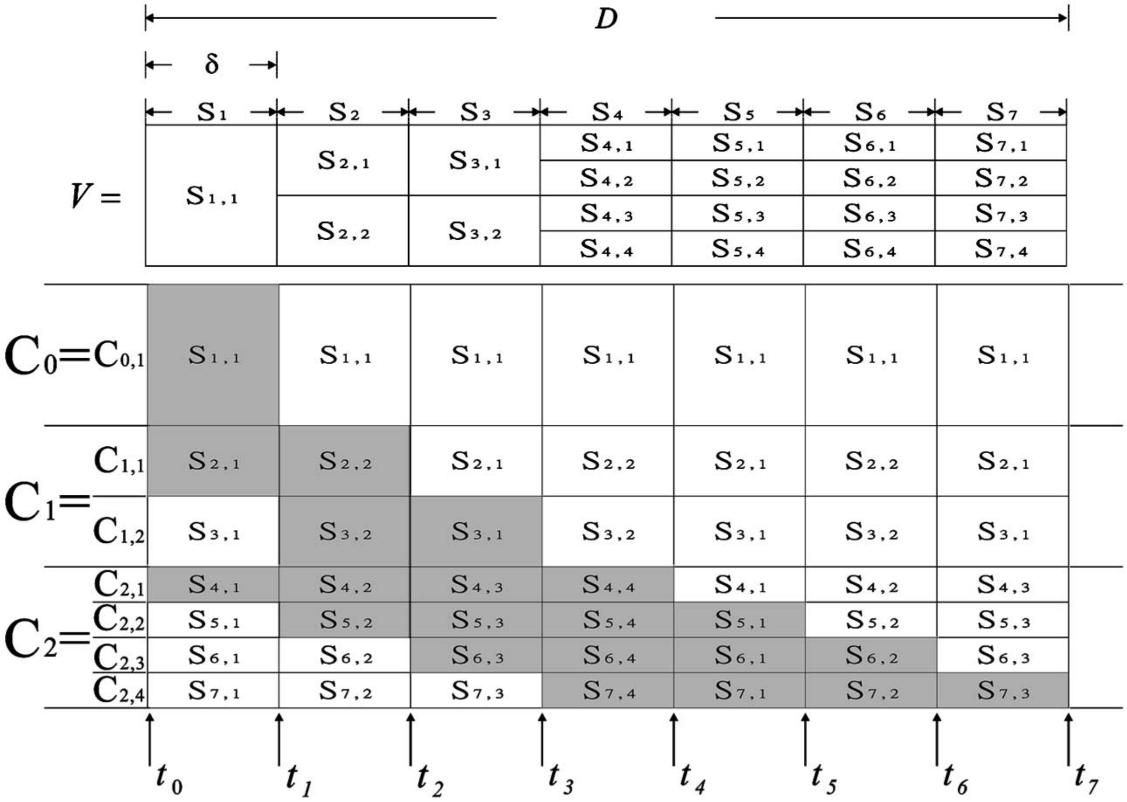

D
δ
S1 S2 S3 S4 S5 S6 S7
V =
S1,1 S2,1 S3,1 S4,1 S5,1 S6,1 S7,1
S4,2 S5,2 S6,2 S7,2
S2,2 S3,2 S4,3 S5,3 S6,3 S7,3
S4,4 S5,4 S6,4 S7,4
C0=C0,1 S1,1 S1,1 S1,1 S1,1 S1,1 S1,1 S1,1
C1= C1,1 S2,1 S2,2 S2,1 S2,2 S2,1 S2,2 S2,1
C1,2 S3,1 S3,2 S3,1 S3,2 S3,1 S3,2 S3,1
C2= C2,1 S4,1 S4,2 S4,3 S4,4 S4,1 S4,2 S4,3
C2,2 S5,1 S5,2 S5,3 S5,4 S5,1 S5,2 S5,3
C2,3 S6,1 S6,2 S6,3 S6,4 S6,1 S6,2 S6,3
C2,4 S7,1 S7,2 S7,3 S7,4 S7,1 S7,2 S7,3
t0 t1 t2 t3 t4 t5 t6 t7

Fig. 2. Video partitioning and scheduling of the SB scheme using $k = 3$ channels.

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 $S_1 \circ S_2 \circ \cdots \circ S_N$.

### B. Staircase Broadcasting (SB) Scheme

The SB scheme [17] improves over the FB scheme on the buffer requirements at the client side while keeping the same viewers' waiting time. It carefully partitions the video both horizontally and vertically. Still, let us consider a video $V$ of length $D$ to be supported by $k$ channels, $C_0, C_1, \ldots, C_{k-1}$. On the server side, the video is broadcast as follows.

1) Vertically partition $V$ into $N$ fixed-length segments, $S_1, S_2, \ldots, S_N$, where $N = 2^k - 1$. The length of each segment is $\delta = (D/N) = (D/2^k - 1)$.
2) Similar to the FB scheme, we will broadcast on $C_i$ the segments $S_{2^i}, S_{2^i+1}, \ldots, S_{2^{i+1}-1}, i = 0 \ldots k-1$, but with the following modification. For each segment $S_j, 2^i \leq j \leq 2^{i+1} - 1$, to be transmitted by $C_i$, we partition it horizontally into $2^i$ sub-segments $S_{j,1}, S_{j,2}, \ldots, S_{j,2^i}$.
3) To cooperate with the above horizontal partitioning, channel $C_i$ is also partitioned horizontally into $2^i$ fixed-bandwidth sub-channels, namely $C_{i,1}, C_{i,2}, \ldots, C_{i,2^i}$. On sub-channel $C_{i,j}, j = 1 \ldots 2^i$, periodically broadcast sub-segments $S_{2^i+j-1,1}, S_{2^i+j-1,2}, \ldots, S_{2^i+j-1,2^i}$.

Fig. 2 shows an example with $k = 3$ channels. On the top, we show that $S_1$ is not partitioned. However, $S_2$ is partitioned into $S_{2,1}$ and $S_{2,2}$, $S_3$ into $S_{3,1}$ and $S_{3,2}$, $S_4$ into $S_{4,1}, S_{4,2}, S_{4,3}$, and $S_{4,4}$, etc. On the bottom, we show that $C_{0,1}$ broadcasts $S_{1,1}$ periodically, $C_{1,1}$ does $S_{2,1}$ and $S_{2,2}$ periodically, $C_{1,2}$ does $S_{3,1}$ and $S_{3,2}$ periodically, $C_{2,1}$ does $S_{4,1}, S_{4,2}, S_{4,3}$, and $S_{4,4}$ periodically, etc.

At the client side, to watch $V$, the following steps are involved.

1) To start the service, wait until the earliest time $t$ when a new time slot begins.
2) Observe that on each sub-channel $C_{i,j}, i = 0 \ldots k-1, j = 1 \ldots 2^i$, the contents being broadcast are the $2^i$ sub-segments of $S_{2^i+j-1}$. The client should grab these sub-segments on $C_{i,j}$ for playback during the time interval $[t + (j-1) \times \delta, t + (2^i + j - 1) \times \delta]$.

For example, in Fig. 2, we show what is to be downloaded for a client starting at time $t_0$. In the first time slot, it will receive segments $S_{1,1}, S_{2,1}$, and $S_{4,1}$ from $C_{0,1}, C_{1,1}$, and $C_{2,1}$, respectively. During this slot, segment $S_{1,1}$ will be consumed, and the other premature segments $S_{2,1}$ and $S_{4,1}$ will be buffered for future use. In the second slot, the client receives segments $S_{2,2}, S_{3,2}, S_{4,2}$, and $S_{5,2}$ from $C_{1,1}, C_{1,2}, C_{2,1}$, and $C_{2,2}$, respectively. Note that during this time slot, the client will combine $S_{2,1}$ in its buffer and $S_{2,2}$ from channel $C_{1,1}$ into segment $S_2$ for its playback. Still, the remaining premature segments $S_{3,2}, S_{5,2}$, and $S_{4,2}$ will be buffered. At the third slot, the client receives segments $S_{3,1}, S_{4,3}, S_{5,3}$, and

$S_{6,3}$ from $C_{1,2}, C_{2,1}, C_{2,2}$, and $C_{2,3}$, respectively. Again, all sub-segments of $S_3$ are available for playback during this slot. This is repeated until the client has consumed all segments.

Intuitively, on each channel $C_i$, we will download $2^i$ sub-segments from each of its sub-channels. The time to start the downloading are the time slot when the corresponding segment is required for playback plus the previous $2^i - 1$ slots. For instance, for segment $S_6$, since it will be played back in time interval $[t_5, t_6]$, we will download its sub-segments in interval $[t_2, t_6]$. It is not hard to prove that the client always has the complete content of the segment to be played back at each time slot.

To summarize, the SB scheme has the same startup latency as the FB scheme, and thus outperforms both the Pyramid scheme [26] and the Permutation-based Pyramid scheme [2] in this regard. Given $k$ channels, a user has to wait no longer than $\delta = (D/2^k - 1)$ time to start the service. The waiting time decreases exponentially as $k$ increases. The average waiting time is $(\delta/2)$. For instance, given a 120-min video, with five channels, viewers have to wait no more than $(120/2^5 - 1) < 4$ minutes to start the service, and with six channels, the maximum waiting time further reduces to $(120/2^6 - 1) < 2$ minutes.

Due to its "staircase" arrangement, the SB scheme delays the reception of some video contents at later time compared to FB, thus saving significant buffering spaces over FB. According to [17], when $k \geq 4$, the buffering requirement of SB is no more than a quarter of the video, as opposed to one half of the video of the FB scheme. In fact, this buffering requirement is almost the same as the Permutation-based Pyramid broadcasting scheme.

Finally, we comment that partitioning a video frame in the horizontal direction may pose difficulties, especially when very fine-grain partitioning is needed. Certain permutation techniques may solve this problem. For example, in Fig. 2, consider the video frames in $S_4$. We can assign the first video frame to $S_{4,1}$, the second video frame to $S_{4,2}$, the third video frame to $S_{4,3}$, the fourth video frame to $S_{4,4}$, and then repeatedly assign the remaining frames to these four sub-segments in a round-robin manner. So no horizontal partitioning is actually needed. However, the cost is an extra delay of four frames time.

## III. Seamless SB Scheme

In this section, we will develop our main result. We will first formulate the seamless channel transition problem. Then we present our modified SB scheme, establish the relationship of channel contents among different channels, and show our channel transition scheme.

### A. *Problem Statement*

Recall that the SB scheme only considers how to support one video given a fixed number, say, $k$, of channels. From a system manager's point of view, it is possible that the number of channels allocated to a video may need to be changed to exploit his/her greatest profit. Suppose that the new number of channels is $k'$. Since customers already starting their services should not experience any disruption during the transition, a naive solution is to allocate another set of $k'$ channels to run the SB scheme. Newly arrived customers are supported by these $k'$ channels. After a while, after all old customers supported by $k$ channels are served, we can release these $k$ channels to the system.

One problem with the above solution is that the transition may take long, thus wasting bandwidth resource. A more serious problem is that the system may not have the extra $k'$ channels, unless the system always spares a sufficient number of channels for the transition needs. This is like the resource deadlock problem in the Operating System design. For example, suppose that we have two videos $V$ and $V'$ supported by $k$ and $k'$ channels, respectively, and we would like to change their numbers of channels to $k'$ and $k$. The transition is impossible without the support of extra resources. Since these are all popular videos, suspending new customers from starting their services during the transition is also infeasible.

With the above observations, we define a channel transition to be *seamless* if the following conditions are satisfied when we change the number of channels for a video $V$ from $k$ to $k'$.

1) Only a total number of $\max\{k, k'\}$ channels are used during the transition.
2) All old customers already starting their services should not experience any disruption for their playback.
3) New customers can arrive at the system in any pattern after the transition point. Also, these new customers are able to start their services by waiting for no longer than the startup latency incurred by using $k'$ channels.
4) In finite amount of time, the system will transit to the normal configuration of using $k'$ channels.

### B. *Transmitting and Receiving Rules*

In this section, we show how to modify the SB scheme to achieve seamless transition. The video $V$ will be partitioned slightly differently. However, the way that segments are broadcast will follow a similar kind of patterns. The reason is that the original SB scheme partitions $V$ into $2^k - 1$ segments given $k$ channels. Since most numbers in $\{2^k - 1 \mid k = 2, 3, 4, \ldots\}$ are mutually prime, finding a proper corresponding relation among segments under different $k$'s is sometimes difficult.

Below, we show the modified SB scheme, given $k$ channels $C_0, C_1, \ldots, C_{k-1}, k \geq 2$.

1) Vertically partition $V$ into $N$ fixed-length segments, $S_1, S_2, \ldots, S_N$, where $N = 3 \times 2^{k-2}$. The length of each segment is $\delta' = (D/N) = (D/3 \times 2^{k-2})$.
2) Broadcast on channel $C_0$ the first segment, $S_1$, periodically. Broadcast on channel $C_1$ the next two segments, $S_2$ and $S_3$, periodically.
3) Broadcast on channel $C_i, i = 2 \ldots k-1$, the segments $S_{3\times 2^{i-2}+1}, S_{3\times 2^{i-2}+2}, \ldots, S_{3\times 2^{i-1}}$, periodically, but with the following arrangement.
   a) For each segment $S_j, 3 \times 2^{i-2} + 1 \leq j \leq 3 \times 2^{i-1}$, to be supported by $C_i$, partition it horizontally into $3 \times 2^{i-2}$ sub-segments, $S_{j,1}, S_{j,2}, \ldots, S_{j,3\times 2^{i-2}}$.
   b) Divide channel $C_i$ horizontally into $3 \times 2^{i-2}$ fixed-bandwidth sub-channels, namely $C_{i,1}, C_{i,2}, \ldots, C_{i,3\times 2^{i-2}}$. On sub-channel $C_{i,j}, j = 1 \ldots 3 \times 2^{i-2}$, periodically broadcast sub-segments $S_{3\times 2^{i-2}+j,1}, S_{3\times 2^{i-2}+j,2}, \ldots, S_{3\times 2^{i-2}+j,3\times 2^{i-2}}$.

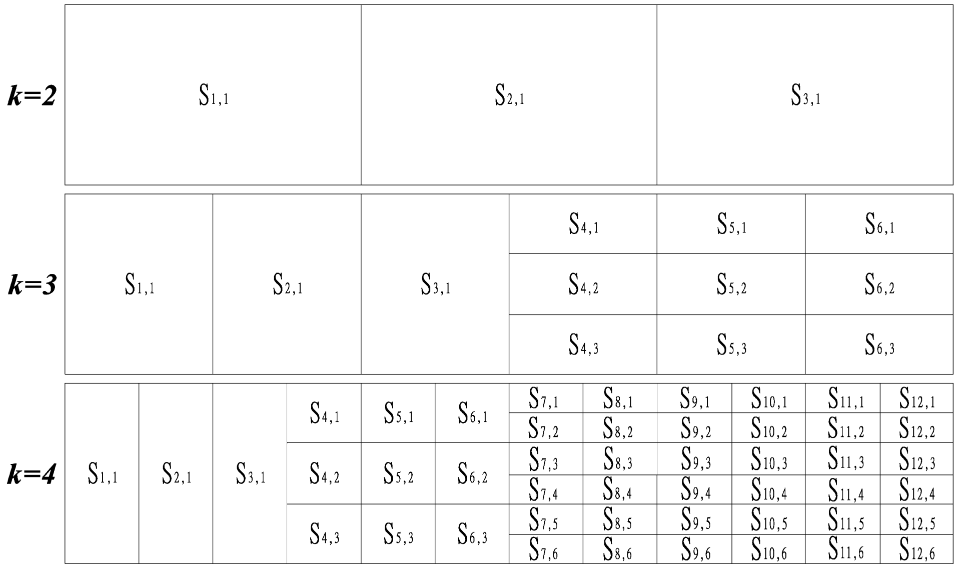

k=2
S1,1
S2,1
S3,1
k=3
S1,1
S2,1
S3,1
S4,1
S5,1
S6,1
S4,2
S5,2
S6,2
S4,3
S5,3
S6,3
k=4
S1,1
S2,1
S3,1
S4,1
S5,1
S6,1
S4,2
S5,2
S6,2
S4,3
S5,3
S6,3
S7,1 S8,1 S9,1 S10,1 S11,1 S12,1
S7,2 S8,2 S9,2 S10,2 S11,2 S12,2
S7,3 S8,3 S9,3 S10,3 S11,3 S12,3
S7,4 S8,4 S9,4 S10,4 S11,4 S12,4
S7,5 S8,5 S9,5 S10,5 S11,5 S12,5
S7,6 S8,6 S9,6 S10,6 S11,6 S12,6

Fig. 3. Vertical and horizontal partitioning of the new seamless SB scheme when using $k = 2, 3, 4$ channels.

For example, Fig. 3 shows how the video is partitioned when using $k = 2, 3$, and 4 channels. For the case of $k = 4$, segments $S_1, S_2, S_3$ are not partitioned horizontally, segments $S_4, S_5, S_6$ are each partitioned horizontally into $3 \times 2^0$ sub-segments to be broadcast on channel $C_2$, and segments $S_7, S_8, \ldots, S_{12}$ are each partitioned horizontally into $3 \times 2^1$ sub-segments to be broadcast on channel $C_3$.

In Fig. 4, we show how video segments are placed on channels based on the above rules when $k = 3$ and 4, respectively.

The receiving rules for the clients are listed below.

1) To start the service, wait until the earliest time $t$ when a new time slot begins.
2) Download the segment on $C_0$ during time interval $[t, t + \delta']$, and the two segments on $C_1$ during time interval $[t, t+ 2\delta']$.
3) Observe that on each sub-channel $C_{i,j}, i = 2 \ldots k - 1, j = 1 \ldots 3 \times 2^{i-2}$, the contents being broadcast are the $3 \times 2^{i-2}$ sub-segments of $S_{3\times 2^{i-2}+j}$. The client should grab these sub-segments on $C_{i,j}$ for playback during the time interval $[t+(j-1)\times\delta', t+(3\times 2^{i-2}+j-1)\times\delta']$.

There is some delicacy in the above design to be noted here. The first three segments $(S_1, S_2, S_3)$ are broadcast in the same way as FB. For the other remaining segments, the clients have to put multiple sub-segments together to construct the original segment. More importantly, each sub-segment is downloaded one time slot *in advance* compared to the original SB. For example, Fig. 4 illustrates how a client starting at time $t_0$ downloads sub-segments (shown in gray) when $k = 3$ and 4. In the case of $k = 4$, the sub-segments $S_{4,1}, S_{4,2}$, and $S_{4,3}$, which together form a full segment $S_4$, are downloaded during $[t_0, t_3]$. But $S_4$ will be consumed by the client during the interval $[t_3, t_4]$. In fact, these sub-segments can be downloaded during $[t_1, t_4]$ (one of the sub-segments will be downloaded "just in time"). The effect is a higher buffering requirement. However, we do this on purpose; the reason will become clear later.

*Theorem 1:* The modified SB scheme guarantees the following properties: 1) a client can play back the video without any disruption, and 2) for each segment $S_i, i \geq 4$, all its sub-segments will already be downloaded in the client's buffer when it is required for playback.

*Proof:* It is easy to see that there will be no disruption for the first three segments. For segment $S_{3\times 2^{i-2}+j}, i = 2 \ldots k - 1, j = 1 \ldots 3 \times 2^{i-2}$, according to the third rule, their contents will be downloaded during the interval $[t+(j-1)\times\delta', t+(3\times 2^{i-2}+j-1)\times\delta']$. It is clear that every consecutive $3 \times 2^{i-2}$ slots render a complete segment $S_{3\times 2^{i-2}+j}$. Since the starting time $t+(j-1)\times\delta' \geq t$ and the finishing time $t+(3\times 2^{i-2}+j-1)\times\delta'$ is right before the time for playback, the client will have all required sub-segments in buffer for playback, hence, the theorem. □

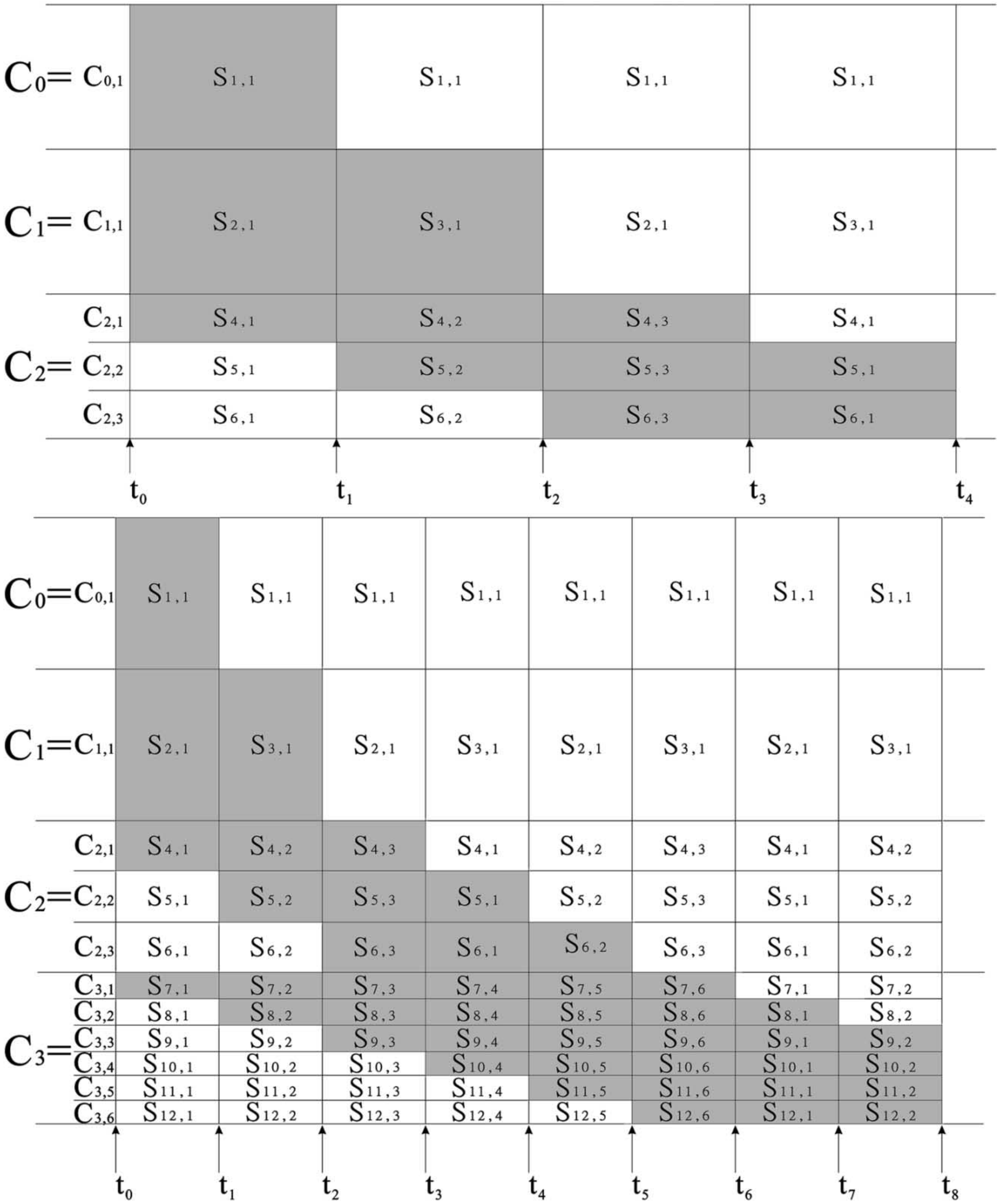
We comment that the second property is essential to guarantee our yet-to-be-presented channel transition to be seamless. Intuitively, this property says that our scheme guarantees that each segment will arrive early enough before the point when its playback starts. As will be clear later, this allows the sub-segments of a segment arrive in arbitrary orders, as long as their arrive time is not beyond the point when playback starts.

### C. Relationship of Channel Contents

Our arrangement does provide some nice relationships between channel contents when using different numbers of channels. This section establishes these relationships, which are necessary to develop our seamless transition result.

Recall that given $k$ channels, we will partition the video into segments $S_i$ and $S_{i,j}$ for some $i$ and $j$. To avoid confusion, below we will denote them by $S_i^k$ and $S_{i,j}^k$, respectively, to indicate that $k$ channels are used. The symbol $\sim$ is used to denote a range of segments; for instance, the segments $S_{i,1}, S_{i,2}, \ldots, S_{i,j}$ can be denoted as $S_{i,1\sim j}$.

Several lemmas are presented below. Note that the discussion will exclude the first three segments $(S_{1,1}, S_{2,1}$, and $S_{3,1})$ since



| | | | | | |
|---|---|---|---|---|---|
| $C_0=$ | $C_{0,1}$ | $S_{1,1}$ | $S_{1,1}$ | $S_{1,1}$ | $S_{1,1}$ |
| $C_1=$ | $C_{1,1}$ | $S_{2,1}$ | $S_{3,1}$ | $S_{2,1}$ | $S_{3,1}$ |
| | $C_{2,1}$ | $S_{4,1}$ | $S_{4,2}$ | $S_{4,3}$ | $S_{4,1}$ |
| $C_2=$ | $C_{2,2}$ | $S_{5,1}$ | $S_{5,2}$ | $S_{5,3}$ | $S_{5,1}$ |
| | $C_{2,3}$ | $S_{6,1}$ | $S_{6,2}$ | $S_{6,3}$ | $S_{6,1}$ |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $C_0=$ | $C_{0,1}$ | $S_{1,1}$ | $S_{1,1}$ | $S_{1,1}$ | $S_{1,1}$ | $S_{1,1}$ | $S_{1,1}$ | $S_{1,1}$ | $S_{1,1}$ |
| $C_1=$ | $C_{1,1}$ | $S_{2,1}$ | $S_{3,1}$ | $S_{2,1}$ | $S_{3,1}$ | $S_{2,1}$ | $S_{3,1}$ | $S_{2,1}$ | $S_{3,1}$ |
| | $C_{2,1}$ | $S_{4,1}$ | $S_{4,2}$ | $S_{4,3}$ | $S_{4,1}$ | $S_{4,2}$ | $S_{4,3}$ | $S_{4,1}$ | $S_{4,2}$ |
| $C_2=$ | $C_{2,2}$ | $S_{5,1}$ | $S_{5,2}$ | $S_{5,3}$ | $S_{5,1}$ | $S_{5,2}$ | $S_{5,3}$ | $S_{5,1}$ | $S_{5,2}$ |
| | $C_{2,3}$ | $S_{6,1}$ | $S_{6,2}$ | $S_{6,3}$ | $S_{6,1}$ | $S_{6,2}$ | $S_{6,3}$ | $S_{6,1}$ | $S_{6,2}$ |
| | $C_{3,1}$ | $S_{7,1}$ | $S_{7,2}$ | $S_{7,3}$ | $S_{7,4}$ | $S_{7,5}$ | $S_{7,6}$ | $S_{7,1}$ | $S_{7,2}$ |
| | $C_{3,2}$ | $S_{8,1}$ | $S_{8,2}$ | $S_{8,3}$ | $S_{8,4}$ | $S_{8,5}$ | $S_{8,6}$ | $S_{8,1}$ | $S_{8,2}$ |
| $C_3=$ | $C_{3,3}$ | $S_{9,1}$ | $S_{9,2}$ | $S_{9,3}$ | $S_{9,4}$ | $S_{9,5}$ | $S_{9,6}$ | $S_{9,1}$ | $S_{9,2}$ |
| | $C_{3,4}$ | $S_{10,1}$ | $S_{10,2}$ | $S_{10,3}$ | $S_{10,4}$ | $S_{10,5}$ | $S_{10,6}$ | $S_{10,1}$ | $S_{10,2}$ |
| | $C_{3,5}$ | $S_{11,1}$ | $S_{11,2}$ | $S_{11,3}$ | $S_{11,4}$ | $S_{11,5}$ | $S_{11,6}$ | $S_{11,1}$ | $S_{11,2}$ |
| | $C_{3,6}$ | $S_{12,1}$ | $S_{12,2}$ | $S_{12,3}$ | $S_{12,4}$ | $S_{12,5}$ | $S_{12,6}$ | $S_{12,1}$ | $S_{12,2}$ |

Fig. 4. Placement of segments in channels in the new seamless SB scheme when using $k = 3$ and 4 channels.

they need special treatment. That is, we will focus only on the contents of the channels $C_2, C_3, \ldots, C_{k-1}$.

*Lemma 1:* Let $k'$ and $k$ be two integers, $k' > k \geq 3$. For any $i$ and $j, 4 \leq i \leq 3 \times 2^{k-2}, 1 \leq j \leq 3 \times 2^{k-3}$, the following equality holds:

$$\begin{aligned} S^{k}_{i,j} = {} & S^{k'}_{(i-1)\cdot 2^{k'-k}+1,(j-1)\cdot 2^{k'-k}+1\sim j\cdot 2^{k'-k}} \\ & \circ S^{k'}_{(i-1)\cdot 2^{k'-k}+2,(j-1)\cdot 2^{k'-k}+1\sim j\cdot 2^{k'-k}} \\ & \circ \cdots \circ S^{k'}_{i\cdot 2^{k'-k},(j-1)\cdot 2^{k'-k}+1\sim j\cdot 2^{k'-k}}. \end{aligned}$$

*Proof:* The proof is by induction on the value of $k' - k$. For the induction basis, when $k' - k = 1$, the lemma can be easily verified, i.e.,

$$S^{k}_{i,j} = S^{k'}_{2i-1,2j-1\sim 2j} \circ S^{k'}_{2i,2j-1\sim 2j}.$$

For example, in Fig. 3, we easily see that $S^{3}_{4,2} = S^{4}_{7,3\sim 4} \circ S^{4}_{8,3\sim 4}$.

For the induction hypothesis, we assume that the lemma is true for $1 \leq k' - k \leq m$. So we have

$$\begin{aligned} S^{k}_{i,j} = {} & S^{k'}_{(i-1)\cdot 2^{m}+1,(j-1)\cdot 2^{m}+1\sim j\cdot 2^{m}} \circ S^{k'}_{(i-1)\cdot 2^{m}+2,(j-1)\cdot 2^{m}+1\sim j\cdot 2^{m}} \\ & \circ \cdots \circ S^{k'}_{i\cdot 2^{m},(j-1)\cdot 2^{m}+1\sim j\cdot 2^{m}}. \end{aligned}$$

The induction basis gives us $S^{k'}_{i,j} = S^{k'+1}_{2i-1,2j-1\sim 2j} \circ S^{k'+1}_{2i,2j-1\sim 2j}$ which leads to

$$\begin{aligned}
S^{k}_{i,j} = &\left(S^{k'+1}_{(2i-1-1)\cdot 2^m+1,(2j-1-1)\cdot 2^m+1\sim 2j\cdot 2^m}\right.\\
&\left.\circ S^{k'+1}_{(2i-1)\cdot 2^m+1,(2j-1)\cdot 2^m+1\sim 2j\cdot 2^m}\right)\\
&\circ \left(S^{k'+1}_{(2i-1-1)\cdot 2^m+2,(2j-1-1)\cdot 2^m+1\sim 2j\cdot 2^m}\right.\\
&\left.\circ S^{k'+1}_{(2i-1)\cdot 2^m+2,(2j-1)\cdot 2^m+1\sim 2j\cdot 2^m}\right)\\
&\circ \cdots \circ \left(S^{k'+1}_{(2i-1)\cdot 2^m,(2j-1-1)\cdot 2^m+1\sim 2j\cdot 2^m}\right.\\
&\left.\circ S^{k'+1}_{(2i)\cdot 2^m,(2j-1)\cdot 2^m+1\sim 2j\cdot 2^m}\right)\\
= &\, S^{k'+1}_{(i-1)\cdot 2^{m+1}+1,(j-1)\cdot 2^{m+1}+1\sim j\cdot 2^{m+1}}\\
&\circ S^{k'+1}_{(i-1)\cdot 2^{m+1}+2,(j-1)\cdot 2^{m+1}+1\sim j\cdot 2^{m+1}}\\
&\circ \cdots \circ S^{k'+1}_{i\cdot 2^{m+1},(j-1)\cdot 2^{m+1}+1\sim j\cdot 2^{m+1}}.
\end{aligned}$$

This establishes the induction step, hence, the lemma. □

While the above lemma establishes the relationship between segments when $V$ is partitioned in different ways, we need to further map them on the time axis to more accurately describe their relationship. Toward this goal, we formulate a channel as an infinite sequence of segments concatenated together and mapped on the time axis $[0, \infty]$.

*Definition 1:* Let $s$ be a video segment. We define $\text{cycle}(s)$ as an infinite, nonstop repetition of $s$ mapped on the time axis $[0, \infty]$ (i.e., $\text{cycle}(s) = s \circ s \circ s \circ \ldots$).

*Definition 2:* Given two integers $k$ and $d$ such that $k > d \geq 2$, define

$$\begin{aligned}
T^k_d = \text{cycle}&\left(S^k_{1+3\cdot 2^{d-2}\sim 3\cdot 2^{d-1},1} \circ S^k_{1+3\cdot 2^{d-2}\sim 3\cdot 2^{d-1},2}\right.\\
&\left.\circ \cdots \circ S^k_{1+3\cdot 2^{d-2}\sim 3\cdot 2^{d-1},3\cdot 2^{d-2}}\right).
\end{aligned}$$

For any integer $j \geq 0$, denote by $T^k_d[j]$ the content of $T^k_d$ during the time interval $[j\delta', (j+1)\delta']$, where $\delta' = (D/3 \cdot 2^{k-2})$.

Intuitively, $T^k_d$ is the video content broadcast on channel $C_d$ when a total of $k$ channels are used. For example, in Fig. 4, when $k = 4$ channels, the contents on channel $C_2$ is $T^4_2 = \text{cycle}(S^4_{4\sim 6,1} \circ S^4_{4\sim 6,2} \circ S^4_{4\sim 6,3})$ if we imagine that the service starts from time 0.

*Definition 3:* Two equal-length video segments $S$ and $S'$ are said to be *similar*, denoted as $S \approx S'$, if their contents are equivalent (not necessarily match exactly in each time instant).

Intuitively, relation "$\approx$" is weaker than "$=$". $S$ and $S'$ are said to be similar if there contents are the same, but may be permuted in time. For example, in Fig. 3, we can claim that $S^3_{1,1} = S^4_{1,1} \circ S^4_{2,1}$ and $S^3_{1,1} \approx S^4_{2,1} \circ S^4_{1,1}$ (the former has exactly the same video content in each time instant, while the latter does not).

The following lemma establishes the channel content relationship when using $k$ channels and using $k+1$ channels. (Note that the $j$th slot of $T^k_d$ is the same interval specified by the $2j$th and $(2j+1)$th slots of $T^{k+1}_{d+1}$.)

*Lemma 2:* Given two integers $k$ and $d$ such that $k > d \geq 2$, the following equality holds for any integer $j \geq 0$:

$$T^k_d[j] \approx T^{k+1}_{d+1}[2j] \circ T^{k+1}_{d+1}[2j+1].$$

*Proof:* By Definition 2 and Lemma 1, we have the equation shown at the bottom of the page. □

For example, in Fig. 3, segments $S^3_{4,3}, S^3_{5,3}$, and $S^3_{6,3}$ at 3 channels are equal to $S^4_{7\sim 12,5}$ and $S^4_{7\sim 12,6}$ at four channels. When mapped to the time axis in Fig. 4, we see that $T^3_2[2] = S^3_{4\sim 6,3} \approx S^4_{7\sim 12,5} \circ S^4_{7\sim 12,6} = T^4_3[4] \circ T^4_3[5]$. Note that some of the sub-segments have been permuted in time.

Next, we extend the above lemma for the scenarios when using $k$ channels and using $k'$ channels such that $k' - k \geq 2$.

*Lemma 3:* Given three integers $k', k$, and $d$ such that $k' > k > d \geq 2$, the following equality holds for any integer $j \geq 0$:

$$\begin{aligned}
T^k_d[j] \approx T^{k'}_{d+k'-k}&[2^{k'-k} \cdot j] \circ T^{k'}_{d+k'-k}[2^{k'-k} \cdot j + 1]\\
&\circ \cdots \circ T^{k'}_{d+k'-k}[2^{k'-k} \cdot j + 2^{k'-k} - 1].
\end{aligned}$$

$$\begin{aligned}
T^k_d[j] &= S^k_{3\times 2^{d-2}+1\sim 3\times 2^{d-1},1+(j \bmod (3\times 2^{d-2}))}\\
&= \left(S^{k+1}_{2(3\times 2^{d-2}+1)-1,2[1+(j \bmod (3\times 2^{d-2}))]-1\sim 2[1+(j \bmod (3\times 2^{d-2}))]}\right.\\
&\quad\left.\circ\ S^{k+1}_{2(3\times 2^{d-2}+1),2[1+(j \bmod (3\times 2^{d-2}))]-1\sim 2[1+(j \bmod (3\times 2^{d-2}))]}\right)\\
&\quad \circ \cdots \circ \left(S^{k+1}_{2(3\times 2^{d-1})-1,2[1+(j \bmod (3\times 2^{d-2}))]-1\sim 2[1+(j \bmod (3\times 2^{d-2}))]}\right.\\
&\quad\left.\circ S^{k+1}_{2(3\times 2^{d-1}),2[1+(j \bmod (3\times 2^{d-2}))]-1\sim 2[1+(j \bmod (3\times 2^{d-2}))]}\right)\\
&\approx S^{k+1}_{2(3\times 2^{d-2}+1)-1\sim 3\times 2^{d},2[1+(j \bmod (3\times 2^{d-2}))]-1}\\
&\quad \circ S^{k+1}_{2(3\times 2^{d-2}+1)-1\sim 3\times 2^{d},2[1+(j \bmod (3\times 2^{d-2}))}]\\
&= S^{k+1}_{3\times 2^{d-1}+1\sim 3\times 2^{d},1+(2j \bmod (3\times 2^{d-1}))}\\
&\quad \circ S^{k+1}_{3\times 2^{d-1}+1\sim 3\times 2^{d},1+((2j+1) \bmod (3\times 2^{d-1}))}\\
&= T^{k+1}_{d+1}[2j] \circ T^{k+1}_{d+1}[2j+1]
\end{aligned}$$

*Proof:* This is proved by repeatedly using Lemma 2 (note that $p = k' - k$)

$$\begin{aligned} T_d^k[j] &\approx T_{d+1}^{k+1}[2j] \circ T_{d+1}^{k+1}[2j+1] \\ &\approx T_{d+2}^{k+2}[4j] \circ T_{d+2}^{k+2}[4j+1] \\ &\quad \circ T_{d+2}^{k+2}[4j+2] \circ T_{d+2}^{k+2}[4j+3] \\ &\quad \cdots \\ &\approx T_{d+p}^{k+p}[2^p j] \circ T_{d+p}^{k+p}[2^p j+1] \\ &\quad \circ \cdots \circ T_{d+p}^{k+p}[2^p j + 2^p - 1]. \end{aligned}$$

□

Intuitively, the above lemma says that the video content in $T_d^k$ during the $j$th time slot can also be seen in $T_{d+k'-k}^{k'}$ during the same period of time, but some sub-segments may be permuted in time. Further, the permutation is bounded within the $j$th time slot. For instance, $T_3^4[2] \approx T_4^5[4] \circ T_4^5[5] \approx T_6^7[16] \circ T_6^7[17] \circ T_6^7[18] \circ T_6^7[19] \circ T_6^7[20] \circ T_6^7[21] \circ T_6^7[22] \circ T_6^7[23]$.

### D. Seamless Channel Transition

In this section, we assume that $V$ is currently supported by $k$ channels. We will show how to seamlessly change the configuration to $k'$ channels (both $k$ and $k' \geq 2$). Let $\Delta k = k' - k$. If $\Delta k > 0$, this is called *positive channel transition* (PCT). If $\Delta k < 0$, this is called *negative channel transition* (NCT).

*1) Positive Channel Transition:* To achieve seamless transition, the basic idea is to guarantee that for any video segment required for playback by an old client, it can be found in the new configuration. We will rely on Lemma 3 to ensure this property. Suppose that the server decides to perform PCT during time slot $t$. The following actions should be taken.

1) At the beginning of time slot $t+1$, do not allow any new clients to start their service. In time slot $t+1$, the server still broadcasts segments based on the same configuration of $k$ channels. At the end of slot $t+1$, release all current $k$ channels.
2) At the beginning of time slot $t+2$, allocate $k'$ channels and start to broadcast segments based on the new configuration of $k'$ channels. At the same time, allow all waiting clients to start their service. Note that since the $(t+2)$th time slot under the old configuration of $k$ channels is the $2^{k'-k} \cdot (t+2)$th time slot under the new configuration of $k'$ channels, the content being broadcast on the new channel $C_d$ after the transition point is $T_d^{k'}[t']$, $d = 2 \ldots k'-1, t' \geq 2^{k'-k} \cdot (t+2)$. (For $C_0$ and $C_1$, we simply repeatedly broadcast $S_{1,1}^{k'}$ and $S_{2,1}^{k'} \circ S_{3,1}^{k'}$, respectively, after the transition point.)

In Fig. 5, we demonstrate how PCT works when $k = 3$ and $k' = 4$. In the example, we assume that it is decided to conduct PCT during slot $t$. New-coming clients during slot $t$ are delayed to be served until the end of slot $t+1$. In fact, according to the above rules, the transition occurs at the end of slot $t+1$. For clients starting at or after the transition point, they download segments based on the new configuration, so there is no problem with their playback. For clients who already started their services before the transition point, we need to prove the transition to be seamless. In Fig. 5, we mark with gray for those slots to be downloaded by clients who start their services at the beginning of slot $t$ (the reader may refer to Fig. 3 for the original video content to prove the correctness).

Below, we prove in general for any $k$ and $k'$ how old clients already starting their playback adapt to the transition.

- For clients starting their service at the beginning of slot $t$, they will download the first three segments $S_{1,1}^k, S_{2,1}^k, S_{3,1}^k$ during slots $t$ and $t+1$. So we can claim that for all clients starting their service before (including) the beginning of slot $t$, no disruption will be experienced for their playback of the first three segments.
- After the transition point, for each segment $T_d^k[j]$ that an old client was supposed to download from the original channel $C_d, d \geq 2$, it will be able to find a similar ($\approx$) content $T_{d+k'-k}^{k'}[2^{k'-k} \cdot j] \circ T_{d+k'-k}^{k'}[2^{k'-k} \cdot j + 1] \circ \cdots \circ T_{d+k'-k}^{k'}[2^{k'-k} \cdot j + 2^{k'-k} - 1]$ from the new channel $C_{d+k'-k}$, according to Lemma 3. Note that "similar" means that the channel content in the slot under consideration may be permuted in time (but bounded within this slot). However, since these contents will only be needed after this time slot (according to the second property of Theorem 1), no disruption will be experienced. So we can claim that the transition is seamless.

*Theorem 2:* The PCT scheme ensures seamless positive transition for all old clients. The transition latency is one time slot, i.e., clients arriving during the slot when we decide to conduct PCT must start at the end of the next slot.

*2) Negative Channel Transition:* NCT is feasible due to the correctness of Lemma 3. Our scheme will gradually release $k - k'$ channels to the system. The scheme does not need any extra channels to support such activity. On deciding to perform NCT, the server takes the following actions.

1) Determine the beginning of the nearest time slot (in terms of the new $k'$-channel configuration). Let the time slot be $t$. (Note that the $t$th time slot under the configuration of $k'$ channels covers the $t \cdot 2^{k-k'}$th, $(t \cdot 2^{k-k'} + 1) - \mathrm{th}, \ldots, ((t+1) \cdot 2^{k-k'} - 1)$th slots under the configuration of $k$ channels.)
2) Starting from the $t$th time slot, periodically broadcast segment $S_{1,1}^{k'}$ on channel $C_0$.
3) Starting from the $(t+1)$th time slot, periodically broadcast segments $S_{2,1}^{k'}$ and $S_{3,1}^{k'}$, in that order, on channel $C_1$.
4) On each channel $C_i, i = 2 \ldots k - k' + 1$, broadcast based on the same schedule as usual under the $k$-channel configuration until the $(t + \lceil(3 \times 2^{i-1} - 2/2^{k-k'})\rceil - 1)$th time slot. Then starting from the $(t + \lceil(3 \times 2^{i-1} - 2/2^{k-k'})\rceil)$th time slot, release $C_i$ to the system.
5) On each channel $C_i, i = k - k' + 2 \ldots k - 1$, broadcast based on the same schedule as usual under the $k$-channel configuration until the $(t + \lceil(3 \times 2^{i-1} - 2/2^{k-k'})\rceil - 1)$th time slot. Then starting from the $(t + \lceil(3 \times 2^{i-1} - 2/2^{k-k'})\rceil)$th time slot, channel $C_i$ is used as channel $C_{i-k+k'}$ under the $k'$-channel configuration (i.e., the video content on the channel will be $T_{i-k+k'}^{k'}$ starting from the $(t + \lceil(3 \times 2^{i-1} - 2/2^{k-k'})\rceil)$th slot).

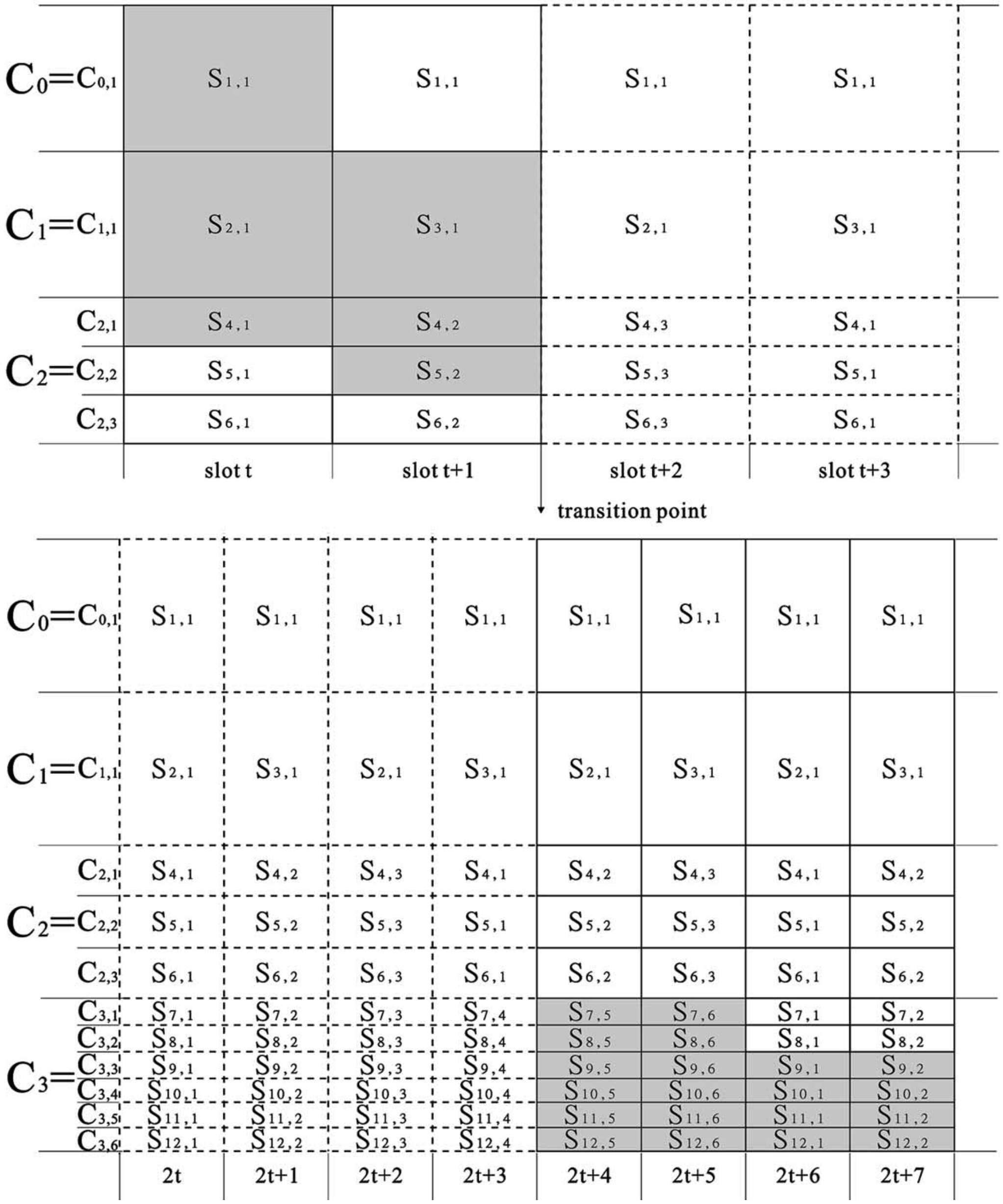

C0=C0,1 S1,1 S1,1 S1,1 S1,1
C1=C1,1 S2,1 S3,1 S2,1 S3,1
C2,1 S4,1 S4,2 S4,3 S4,1
C2=C2,2 S5,1 S5,2 S5,3 S5,1
C2,3 S6,1 S6,2 S6,3 S6,1
slot t
slot t+1
slot t+2
slot t+3
transition point
C0=C0,1 S1,1 S1,1 S1,1 S1,1 S1,1 S1,1 S1,1 S1,1
C1=C1,1 S2,1 S3,1 S2,1 S3,1 S2,1 S3,1 S2,1 S3,1
C2,1 S4,1 S4,2 S4,3 S4,1 S4,2 S4,3 S4,1 S4,2
C2=C2,2 S5,1 S5,2 S5,3 S5,1 S5,2 S5,3 S5,1 S5,2
C2,3 S6,1 S6,2 S6,3 S6,1 S6,2 S6,3 S6,1 S6,2
C3,1 S7,1 S7,2 S7,3 S7,4 S7,5 S7,6 S7,1 S7,2
C3,2 S8,1 S8,2 S8,3 S8,4 S8,5 S8,6 S8,1 S8,2
C3= C3,3 S9,1 S9,2 S9,3 S9,4 S9,5 S9,6 S9,1 S9,2
C3,4 S10,1 S10,2 S10,3 S10,4 S10,5 S10,6 S10,1 S10,2
C3,5 S11,1 S11,2 S11,3 S11,4 S11,5 S11,6 S11,1 S11,2
C3,6 S12,1 S12,2 S12,3 S12,4 S12,5 S12,6 S12,1 S12,2
2t
2t+1
2t+2
2t+3
2t+4
2t+5
2t+6
2t+7

Fig. 5. PCT example with $k = 3$ channels and $k' = 4$ channels. The dotted segments are shown for ease of understanding; they are not actually transmitted on the channels.

Fig. 6 shows how to transit from $k = 4$ channels to $k' = 3$ channels. The transition point is at the beginning of the $2t$th slot under the old configuration, or the beginning of the $t$th slot under the new configuration. Channels $C_0$ and $C_1$ start to periodically broadcast $S^3_{1,1}$ and $S^3_{2,1} \circ S^3_{3,1}$ from $t$ and $t+1$ slots, respectively. Channel $C_2$ will broadcast as usual until the end of slot $t+1$, and then it can be released. Channel $C_3$ will broadcast as usual until the end of slot $t + 4$, and then it will start to broadcast based on the new configuration (i.e., $T^3_2$). The gray segments are what is to be consumed by the latest clients under the old configuration (starting service at slot $2t - 1$).

To summarize, old clients starting before the transition point will not experience disruption because we always broadcast on all channels all required segments under the old configuration as usual, until the related contents are not needed by any old client (rules 4 and 5 are designed to guarantee this). For new clients, care must be taken. Channel $C_0$ switches to the new $k'$-channel configuration immediately, because its content is not needed by any old client. Similarly, channel $C_1$ switches to the new configuration after one slot, because after that point its content is not needed by any old client. It is not hard to observe that new clients starting at any time can download the first three segments $S^{k'}_{1,1}$, $S^{k'}_{2,1}$, and $S^{k'}_{3,1}$ for playback without disruption.

For the remaining channels, Lemma 3 guarantees an important mapping from each channel $C_i, i = k - k' + 2 \ldots k - 1$, under the $k$-channel configuration to each channel $C_{i-k+k'}$ under the new $k'$-channel configuration. Specifically, in each time slot, the content that can be found in the new configuration can also be found in the old configuration, but some subsegments may be permuted in time, bounded within the same slot. However, the permutation will not cause any problem because the related content will not be needed for playback before

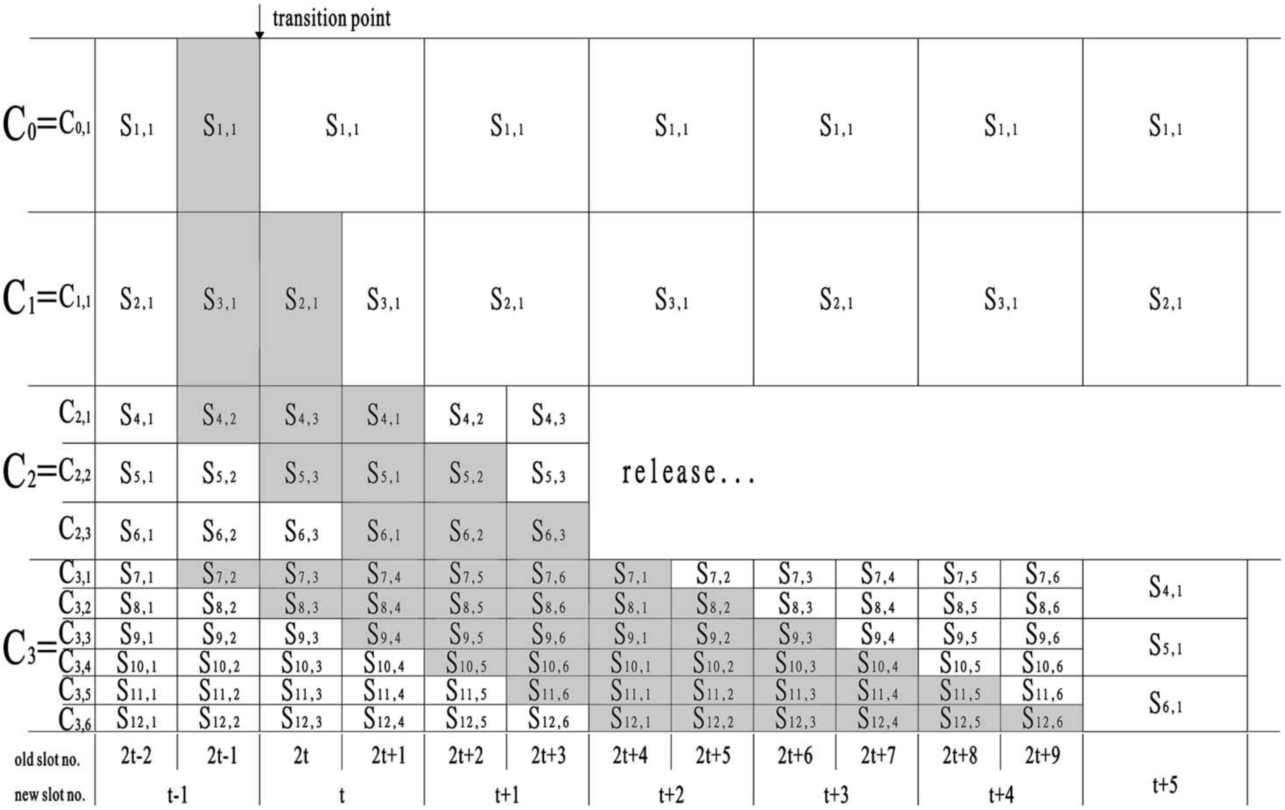

transition point
C0=C0,1
S1,1 S1,1 S1,1 S1,1 S1,1 S1,1 S1,1 S1,1
C1=C1,1
S2,1 S3,1 S2,1 S3,1 S2,1 S3,1 S2,1 S3,1 S2,1
C2,1
S4,1 S4,2 S4,3 S4,1 S4,2 S4,3
C2=C2,2
S5,1 S5,2 S5,3 S5,1 S5,2 S5,3
release...
C2,3
S6,1 S6,2 S6,3 S6,1 S6,2 S6,3
C3=
C3,1
S7,1 S7,2 S7,3 S7,4 S7,5 S7,6 S7,1 S7,2 S7,3 S7,4 S7,5 S7,6
C3,2
S8,1 S8,2 S8,3 S8,4 S8,5 S8,6 S8,1 S8,2 S8,3 S8,4 S8,5 S8,6
S4,1
C3,3
S9,1 S9,2 S9,3 S9,4 S9,5 S9,6 S9,1 S9,2 S9,3 S9,4 S9,5 S9,6
C3,4
S10,1 S10,2 S10,3 S10,4 S10,5 S10,6 S10,1 S10,2 S10,3 S10,4 S10,5 S10,6
S5,1
C3,5
S11,1 S11,2 S11,3 S11,4 S11,5 S11,6 S11,1 S11,2 S11,3 S11,4 S11,5 S11,6
C3,6
S12,1 S12,2 S12,3 S12,4 S12,5 S12,6 S12,1 S12,2 S12,3 S12,4 S12,5 S12,6
S6,1
old slot no.
2t-2 2t-1 2t 2t+1 2t+2 2t+3 2t+4 2t+5 2t+6 2t+7 2t+8 2t+9
new slot no.
t-1 t t+1 t+2 t+3 t+4 t+5

Fig. 6. NCT example with $k = 4$ channels and $k' = 3$ channels.

TABLE II
STARTUP LATENCIES FOR A 120-MIN VIDEO USING $k$ CHANNELS IN DIFFERENT SCHEMES

| $k$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| SSB, SFB ($\alpha = 2$) | 2400 | 1200 | 600 | 300 | 150 | 75 | 37.5 | 18.75 |
| SFB ($\alpha = 3$) | | 1028.571 | 514.285 | 257.142 | 128.571 | 64.285 | 32.142 | 16.071 |
| SFB ($\alpha = 4$) | | | 480 | 240 | 120 | 60 | 30 | 15 |
| SFB ($\alpha = 5$) | | | | 232.258 | 116.129 | 58.064 | 29.032 | 14.516 |
| FB, SB | 2400 | 1028.571 | 480 | 232.258 | 114.285 | 56.692 | 28.235 | 14.09 |

the end of the time slot (guaranteed by Theorem 1). Thus, for a new client arriving after the transition point, it must download based on the old configuration before (including) the $(t + \lceil(3 \times 2^{i-1} - 2/2^{k-k'})\rceil - 1)$th slot on $C_i$, and download based on the new configuration after that slot.

Channels in the middle will be released to the system. For example, if we transit from $k = 15$ channels to $k' = 6$ channels, channels $C_2, C_3, \ldots, C_{10}$ under the old configuration will be released, and the remaining channels $C_{11}, C_{12}, C_{13}$, and $C_{14}$ will be used as channels $C_2, C_3, C_4$, and $C_5$, respectively, under the new configuration.

## IV. PERFORMANCE COMPARISON

In this section, we compare our SSB scheme to the seamless FB (SFB) scheme and other schemes without seamless transition capability from several aspects.

### A. Startup Latency

Assuming that the video size is $D$, the original SB scheme has a slot length of $(D/2^k - 1)$. The modified SSB has a slot length of $(D/3 \times 2^{k-2})$. The increase is about a factor of $1/3$, but the absolute amount of increase will converge quickly to zero as $k$ increases. In Table II, we compare the maximum startup latencies for a 120-min video using $k$ channels in different schemes. Note that the latency of SFB depends on a system parameter $\alpha$, which is the least number of channels that SFB requires to guarantee seamless transition [25].

### B. Buffering Space

When $k \leq 2$, it is easy to derive the buffering requirement. The next theorem derives the buffering requirement when $k \geq 3$.

*Theorem 3:* When $k \geq 3$, the SSB scheme requires to buffer at most $((1/4) + (1/3 \times 2^{k-1}))D$ of the video.

*Proof:* From the pattern that a client pre-stores sub-segments, we observe that the buffering volume will reach the

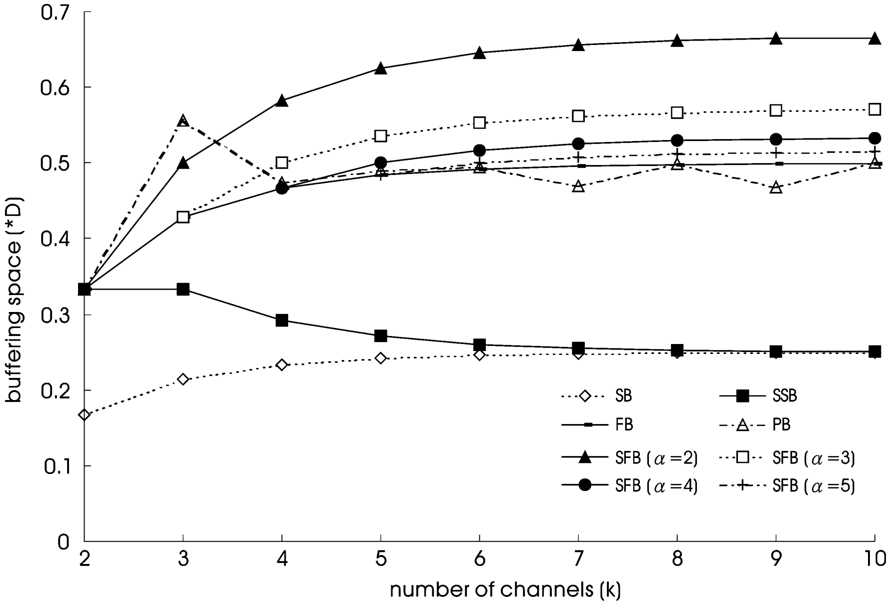


Fig. 7. Comparison of buffering requirements.

highest at the moment when the last sub-segment in channel $C_{k-2}$ is buffered (for example, when $k = 4$, this is time $t_5$ in Fig. 4). At this moment, the following segments are in buffer:

- the complete segment $S_{3\times 2^{k-3}}$ downloaded from $C_{k-2}$;
- $3\times 2^{k-3}-1$ sub-segments downloaded from sub-channel $C_{k-1,1}$, $3\times 2^{k-3}-2$ sub-segments downloaded from sub-channel $C_{k-1,2}, \ldots$, and 1 sub-segment downloaded from sub-channel $C_{k-1,3\times 2^{k-3}-1}$.

Summing all these together, the maximum buffering requirement is (in terms of video length)

$$\left(1+\frac{1}{3\times 2^{k-3}}\times\left(1+2+\cdots+(3\times 2^{k-3}-1)\right)\right)\times\frac{D}{3\times 2^{k-2}}=\left(\frac{1}{4}+\frac{1}{3\times 2^{k-1}}\right)\times D.$$

□

The maximum buffering requirement of the original SB is $(2^{k-1}-1)/(2^{k+1}-2)\times D$. The amount of increase will converge to zero as $k$ increases. In Fig. 7, we compare the buffering requirements of different schemes. Basically, the SB-based schemes' buffering space is about one half of the FB-based schemes. This shows that our SSB, while achieving seamless channel transition, does maintain the original SB's merit of small buffering requirement.

### *C. Channel Release Time*

In NCT, channels will not be released immediately after the transition point. If we perform NCT from $k$ channels to $k'$ channels, it will take $(3\times 2^{i-1}-2)/2^{k-k'}$ time slots to release channel $C_i, i = 2\ldots k-k'+1$ ("time slot" in terms of the new configuration). In Fig. 8, we compare the channel release time of SSB and SFB when transiting from $k+1$ channels to $k$ channels. Our release time is longer because the philosophy of the SB scheme is to download segments in a "lazier" manner than the FB scheme so as to reduce buffering requirement.

## V. Channel Allocation Policy

In the above, we only address the behavior for one particular video. Given a set of popular videos each with a certain level of demand, it would be desirable to know how to distribute the available channels to each video from a system manager's viewpoint. We formulate this problem as one of minimizing the average waiting time incurred over all viewers, given a global buffering space constraint for all viewers. Suppose that the system has in total $K$ channels to support $m$ popular videos $V_1, V_2, \ldots, V_m$ of lengths $D_1, D_2, \ldots, D_m$, respectively. Let $\lambda_i$ be the request arrival rate of $V_i, i = 1\ldots m$, and $k_i$ the number of channels assigned to $V_i$. Our goal is minimize

$$\sum_{i=1}^{m}\lambda_i * \frac{D_i}{3\cdot 2^{k_i-2}} \tag{1}$$

subject to the constraints

$$\sum_{i=1}^{m}k_i\le K,$$
$$\frac{D_i}{4}+\frac{D_i}{3\cdot 2^{k_i-1}}\le B$$

where $B$ is a fixed hardware buffering constraint for all set-top boxes.

We first simplify the two constraints to get a clearer picture. The first constraint can be changed $\sum_{i=1}^{m}k_i = K$, since increasing channels to any video can always reduce the waiting time of that particular video. Further, according to Fig. 7, the buffering requirement is a strictly decreasing value as the number of channels increases. So the second constraint reduces to a simpler form of

$$k_i\ge 1+\log_2\frac{D_i}{3\cdot\left(B-\frac{D_i}{4}\right)}=b_i.$$

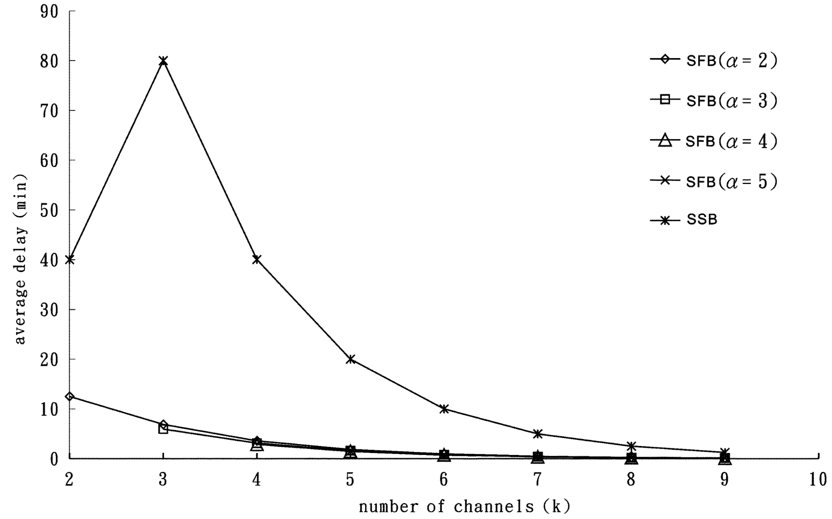


Fig. 8. Channel release time of NCT when transiting from $k+1$ channels to $k$ channels.

Note that for each video $V_i, b_i$ is a constant, which sets a lower bound for $k_i$. It follows that the total number of available channels, $K$, in the system is at least $\sum_{i=1}^{m} b_i$. Otherwise, at least one of the videos can not be supported by SSB without violating the buffering constraint. As a result, we always assume $K \geq \sum_{i=1}^{m} b_i$.

Below, we propose a greedy approach to minimize (1). First, to satisfy the buffering constraint, each $V_i$ is given $b_i$ channels, $i = 1 \ldots m$. For each of the remaining $K - \sum_{i=1}^{m} b_i$ channels, we greedily assign it to the video which leads to the maximum reduction in (1). Note that the reduction can be easily calculated since an increase of one channel to a video will reduce its current waiting time by half. This is repeated until all channels are assigned.

*Theorem 4:* The above greedy approach guarantees the minimal average waiting time over all viewers.

*Proof:* For each $V_i$, after the initial assignment of $b_i$ channels, its cost factor is $(\lambda_i D_i)/(3 \cdot 2^{k_i - 2})$. If we keep on increasing channels for it, the expression is halved after each increase. Now we list the sequence of reductions that may be obtained for each $V_i$, if an infinite number of channels are available:

$$\begin{aligned} V_1 &: \frac{\lambda_1 D_1}{3 \cdot 2^{k_1 - 1}}, \frac{\lambda_1 D_1}{3 \cdot 2^{k_1}}, \frac{\lambda_1 D_1}{3 \cdot 2^{k_1 + 1}}, \ldots \\ V_2 &: \frac{\lambda_2 D_2}{3 \cdot 2^{k_2 - 1}}, \frac{\lambda_2 D_2}{3 \cdot 2^{k_2}}, \frac{\lambda_2 D_2}{3 \cdot 2^{k_2 + 1}}, \ldots \\ &\cdots \\ V_m &: \frac{\lambda_m D_m}{3 \cdot 2^{k_m - 1}}, \frac{\lambda_m D_m}{3 \cdot 2^{k_m}}, \frac{\lambda_j D_m}{3 \cdot 2^{k_m + 1}}, \ldots. \end{aligned}$$

Minimizing (1) is equivalent to maximizing the total amount of reductions in the above sequences. So the problem becomes choosing in total $K - \sum_{i=1}^{m} b_i$ numbers from the above sequences, each from left to right, such that their sum is maximal. Since each sequence is strictly descending, it is easy to see that the proposed greedy approach will achieve this goal. □

## VI. Conclusion

In this paper, we have developed a seamless channel transition scheme based on the SB scheme. With the modified SB scheme, the startup latency is only increased slightly as opposed to the original SB scheme, and the advantage of low buffering requirement of the original SB scheme is maintained. The saved memory space may reduce the hardware costs set-top boxes or may be used for other purposes (such as data communication). Moreover, the modified scheme has conquered the problem of dynamically and adaptively changing the number of channels being used, which is impossible in the original SB scheme. To our knowledge, only the FB and SB schemes have variations with seamless channel transition capability. A generalized methodology to this problem (for different broadcasting schemes) is definitely desirable, but could be very difficult to achieve.

## References

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

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

[3] L. Atzori, F. D. Natale, M. D. Gregorio, and D. D. Giusto, "Multimedia information broadcasting using digital TV channels," *IEEE Trans. Broadcasting*, vol. 43, pp. 242–251, Sept. 1997.

[4] A. Bar-Noy, J. Goshi, R. E. Ladner, and K. Tarn, "Comparison of stream merging algorithms for media-on-demand," in *SPIE Conf. Multimedia Computing and Networking*, 2002, pp. 115–129.

[5] 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.

[6] T. Chiueh and C. Lu, "A periodic broadcasting approach to video-on-demand service," *Proc. SPIE*, vol. 2615, pp. 162–169, Oct. 1995.

[7] M. Cominetti, V. Mignone, A. Morello, and M. Visintin, "The european system for digital multi-programme television by satellite," *IEEE Trans. Broadcasting*, vol. 41, pp. 49–62, June 1995.

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

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

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

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

[12] L. Gao and D. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," *IEEE Multimedia*, vol. 2, pp. 117–121, June 1999.

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

[14] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," *Proc. ACM SIGCOMM*, pp. 89–100, Sept. 1997.

[15] L.-S. Juhn and L.-M. Tseng, "Fast broadcasting for hot video access," *Proc. Real-Time and Embedded Computing Systems and Applications Conf. (RTCSA)*, pp. 237–243, Oct. 1997.

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

[17] L.-S. Juhn and L.-M. Tseng, "Staircase data broadcasting and receiving scheme for hot video service," *IEEE Trans. Consumer Electron.*, vol. 43, pp. 1110–1117, Nov. 1997.

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

[19] L.-S. Juhn and L.-M. Tseng, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Trans. Broadcasting*, vol. 44, pp. 100–105, Mar. 1998.

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

[21] A. Mahanti, E. L. Eager, M. K. Vernon, and D. Sundaram-Stukel, "Scalable on-demand media streaming with packet loss recovery," in *Proc. ACM SIGCOMM*, 2001, pp. 195–209.

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

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

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

[25] Y. C. Tseng, C. M. Hsieh, M. H. Yang, W. H. Liao, and J. P. Sheu, "Data broadcasting and seamless channel transition for highly-demanded videos," in *Proc. IEEE INFOCOM*, 2000, pp. 727–736.

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

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

**Yu-Chee Tseng** (M'95–SM'03) received the B.S. degree from the National Taiwan University, Taiwan, R.O.C., in 1985 and the M.S. degree from National Tsing-Hua University, Taiwan, in 1987, both in computer science. He received 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 in the Department of Computer Science, Chung-Hua University. He joined the Department of Computer Science and Information Engineering, National Central University, in 1996, and became a Full Professor in 1999. Since August 2000, he has been a Full Professor in the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan. His research interests include mobile computing, wireless communication, network security, and parallel and distributed computing.

Dr. Tseng has served as a Program Chair in the Wireless Networks and Mobile Computing Workshop, 2000 and 2001, as an Associate Editor for *The Computer Journal*, as a Guest Editor for *ACM Wireless Networks* special issue on Advances in Mobile and Wireless Systems, as a Guest Editor for IEEE TRANSACTIONS ON COMPUTERS special issue on Wireless Internet, as a Guest Editor for the *Journal of Internet Technology* special issue on Wireless Internet: Applications and Systems, as a Guest Editor for *Wireless Communications and Mobile Computing* special issue on Research in Ad Hoc Networking, Smart Sensing, and Pervasive Computing, as an Editor for the *Journal of Information Science and Engineering*, and as a Guest Editor for *Telecommunication Systems* special issue on Wireless Sensor Networks. He received the Outstanding Research Award, 2001–2002, from the National Science Council, R.O.C.

**Yu-Chi Chueh** received the B.S. degree in computer science from Tamkang University, Tamsui, Taiwan, R.O.C., in 1999, and the M.S. degree in computer science from the National Central University, Chung-Li, Taiwan, in 2001.

His research interests include video-on-demand and interactive multimedia.

**Jang-Ping Sheu** (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 the National Tsing Hua University, Taiwan, in 1983 and 1987, respectively.

He joined the faculty of the Department of Electrical Engineering, National Central University, Taiwan, as an Associate Professor in 1987. He is currently a Professor of the Department of Computer Science and Information Engineering, National Central University. He was a Chair of Department of Computer Science and Information Engineering, National Central University from August 1997 to July 1999. He was a Visiting Professor at the Department of Electrical and Computer Engineering, University of California, Irvine, from July 1999 to April 2000. His current research interests include wireless communications, mobile computing, parallel processing, and distributed computing Systems. He was an Associate Editor of the *Journal of the Chinese Institute of Electrical Engineering*, from August 1996 to July 2000. He was an Associate Editor of the *Journal of Information Science and Engineering* from August 1996 to July 2002. He is an Associate Editor of the *Journal of the Chinese Institute of Engineers*. He was a Guest Editor of a special issue of the *Wireless Communications and Mobile Computing Journal*. He was a Program Chair of IEEE ICPADS'2002 and a Vice-Program Chair of ICPP 2003.

Dr. Sheu received the Distinguished Research Awards of the National Science Council of the Republic of China for 1993–1994, 1995–1996, and 1997–1998. He was the Specially Granted Researcher, National Science Council, from 1999 to 2002. He received the Distinguished Engineering Professor Award of the Chinese Institute of Engineers in 2003. Dr. Sheu is a member of the Association for Computing Machinery and the Phi Tau Phi Society.