Circuits and Systems, **2011, 2, 151-161**
doi:10.4236/cs.2011.23023 Published Online July 2011 (http://www.SciRP.org/journal/cs)

Scientific Research

Adaptability of Conservative Staircase Scheme for Live Videos

Sudeep Kanav, Satish Chand
Division of Computer Engineering, *Netaji Subhas Institute of Technology*, *New Delhi*, *India*
E-mail: *sudeepkanav@gmail.com*, *schand*86*@hotmail.com*
Received August 8, 2010; *revised April* 18, 2011; *accepted April* 25, 2011

Abstract

Existing broadcasting schemes provide services for the stored videos. The basic approach in these schemes is to divide the video into segments and organize them over the channels for proper transmission. Some schemes use segments as a basic unit, whereas the others require segments to be further divided into subsegments. In a scheme, the number of segments/subsegments depends upon the bandwidth allocated to the video by the video server. For constructing segments, the video length should be known. If it is unknown, then the segments cannot be constructed and hence the scheme cannot be applied to provide the video services. This is an important issue especially in live broadcasting applications wherein the ending time of the video is unknown, for example, cricket match. In this paper, we propose a mechanism for the conservative staircase scheme so that it can support live video broadcasting.

Keywords: Conservative Staircase Scheme, Staircase Scheme, Live Video Channel, Channel Transition

1. Introduction

Video broadcasting has been an active research area for last few years and several broadcasting schemes have been developed. The technologies available earlier for these applications could not support high data rate and hence the video services could not gain popularity in spite of their vast applications. Besides the high data rate, their storage requirement is also quite high unless some compression technique is applied. In fact, even after applying a good compression technique the data size is considerably large. In recent years, the communication and computational technologies (including the storage technologies) have been developed significantly. But new applications such as Video-on-Demand (VOD) put a limitation on data rate and the storage devices. So, these resources need to utilize efficiently. Several good schemes have been developed to provide the video services. In almost all the schemes, the video data is transmitted in terms of segments and/or subsegments and the size of a segment and/or subsegment is determined based on the bandwidth allocated to the video. For applying a broadcasting scheme, the video size should be known. In case of live videos, the size of the video object is not known in the beginning and thus the schemes cannot be employed to provide the live video services.

There are generally two main approaches for providing video services. In the first approach, the bandwidth is allocated to the individual users and in the second one the bandwidth is allocated to the individual video objects. In the first case, the immediate video services are provided to user requests and the number of users is the main constraint. In the second case, the video services are independent of the number of users, but all users may not get immediate services. The first approach is called user-centered or true video-on-demand and the second one is called data-centered or near video-on-demand. In both the approaches, the video server is one of the very important entities, which allocates bandwidth to videos. The bandwidth is a scarce resource and must be used efficiently. For allocating bandwidth to the video objects, many researchers have discussed several schemes [1-6] and all these schemes are meant for the stored videos. To the best of authors' knowledge, there does not appear any work that discusses the live video transmission. The possible reason might be the unknown video size in advance as all schemes require constructing the segments/subsegments from the video. To develop a broadcasting scheme to support live video broadcasting is the motivation to carry out this work. In this paper, we propose a technique that makes the conservative staircase scheme to provide live video broadcasting.

The system design consists of a live system that broadcasts the live video using its live video channel. Besides the live system, it contains a video server that stores video data from the live system into its buffer and then broadcasts that data. Storing video data from the live system by the video server is done in terms of pre-specified fixed size durations. We call such durations as time slots and the data downloaded in a time slot is referred to as a segment. The segment size (in time units) determines the user's waiting time. The live system just broadcasts the live video; it does not store. The video server while broadcasting the stored video data from its buffer downloads new data from the live system into its buffer in terms of segments. The new stored segments are broadcast by the video server along with the old segments. This process continues till the live video transmission is there. When the live video broadcasting is over, the video size becomes known and the scheme can function similar to a scheme meant for the video of known size. If the live broadcast continues and all video channels of the video server have been exhausted**,** then the newly downloaded segments cannot be broadcast. Therefore, we need to make some video channel free for broadcasting the new segments. This can be done if the data occupied by a channel is moved to other channels. While carrying out this activity, all users must get reliable services. To transfer data from one channel to another without interrupting user services is called channel transition. So, we need to apply channel transition mechanism to make the last channel free by transferring its data to other video channels. Thus, the newly downloaded segments can be broadcast using this free channel. This is the concept used in this paper.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 discusses architecture of the scheme for live video transmission. Section 4 presents the results and discussion. Finally, in Section 5 the paper is concluded.

2. Related Work

Several broadcasting schemes have been discussed in literature. Some of the important schemes are harmonic scheme [7], cautious harmonic scheme [8], skyscraper scheme [9], and conservative staircase scheme [10]. The harmonic and cautious harmonic schemes perform better than the skyscraper and conservative staircase schemes, but their implementation is more complex. These schemes use non-uniformly allocated bandwidth logical channels. The skyscraper and conservative staircase schemes use uniformly allocated bandwidth logical channels, called video channels, which are individually divided into uniform subchannels. A subchannel transmits a segment in terms its subsegments. In this paper, we will refer the conservative staircase scheme as the conservative scheme.

In almost all the schemes, the first one or two channels are generally kept undivided and other channels may be divided into subchannels. All these channels are generally video channels. A logical channel with bandwidth equal to the consumption rate of the video is called the *video channel*. The video is divided into equal-sized segments; each segment may further be individually divided into uniform subsegments. The conservative scheme has been developed to overcome the limitation of the staircase scheme [11]. The problem with the staircase scheme is that this scheme does not always provide the video data to all users in time. The staircase scheme has been developed to overcome the excessive buffer requirement of the Fast Broadcasting scheme [12] without increasing the user's waiting time. The proposed scheme is based upon the conservative staircase scheme [10]. So, we briefly review this scheme. In the conservative scheme, the video is uniformly divided into segments and the bandwidth allocated to the video into uniform channels. The video display time is divided into equal time durations, called time slots. The segment size (in time units) is equal to the time slot length. More precisely, a time slot is the duration in which a segment can be viewed exactly at the consumption rate. Let the number of segments of a video of length D be K ($K \geq 3$), denoting them as S_1, S_2,⋯, S_K, and the video channels allocated to it be N. The number of video segments and the number of video channels are related by $K = 3 * 2^{N-2}$. The first segment S_1 is transmitted over the first video channel. The next two segments are transmitted over the second video channel. The segments from $(1 + 3*2^{m-3})^{\text{th}}$ to $(3*2^{m-2})^{\text{th}}$ are transmitted over mth video channel C_m ($m > 2$). For transmitting data over the mth channel, this channel is divided into $3*2^{m-3}$ number of subchannels and the corresponding segments are divided into subsegments. The subsegment $S_{i,j}$ ($3*2^{m-3} < i \leq 3*2^{m-2}$, $m > 2$) is transmitted over the jth subchannel of the mth channel in $(p*3*2^{m-3} + (i + j - 1) \bmod 3*2^{m-3})^{\text{th}}$ time slot, $p = 0,1,2,\cdots$. **Figure 1** shows transmission of the video segments and subsegments over three video channels in the conservative scheme.

The conservative scheme overcomes the limitation of the staircase scheme. In the staircase scheme, all users may not get video data on time. However, the conservative scheme requires more bandwidth as compared to the staircase scheme. Since the conservative scheme is complete in itself, *i.e.*, it provides video data to all users on time, we consider it to support live videos and this is the main contents of this paper. In [13], the live broadcasting mechanism has been discussed for the Fast broadcasting scheme, but the Fast broadcasting scheme requires quite

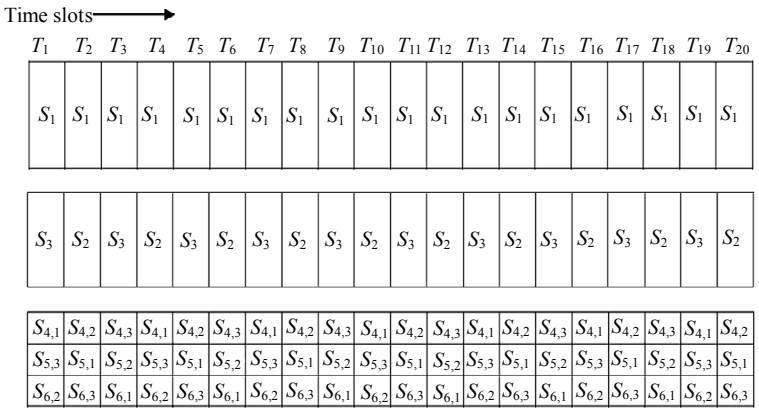

Figure 1. Transmission of segments/subsegments in conservative scheme.

large amount of storage. That is why we consider the conservative staircase scheme for live broadcasting. In next section, we discuss the proposed scheme.

3. Adaptability of Conservative Scheme for Live Videos

The conservative scheme needs the video size in the beginning to partition it into equal-sized segments and subsegments. Generally, in a live video we do not know the video size; so this scheme cannot be applied in its existing form. We modify its basic architecture. We do not divide the segments or video channels any further. For the modified conservative scheme, we discuss a mechanism so that this scheme can support live video broadcasting. We assume that the bandwidth allocated to the video is finite. This assumption is not illogical because for abundant bandwidth there is hardly any issue to discuss.

In the proposed architecture, we have a *live* system that broadcasts the live video using its video channel, called the *live channel*. This live system is active while there is a live video and provides video services only once using its live video channel. There is one more system that stores the video data from the live system into its buffer. This system, called video server, broadcasts the stored video data. The live system broadcasts the video data at the consumption rate. The user requests received till the live system begins to broadcast the video data get all data from the live system directly. These requests require no buffer storage. The live video display is divided into fixed time durations, called *time slots*. The video server stores the video data from the live channel. The data downloaded and stored in a time slot constitutes a video segment. The segment size (in time units) determines the user's waiting time. After storing new segment in its buffer, the video server broadcasts that segment over its video channels and concurrently downloads new segments from the live system into its buffer. A request received after the live system has started the live video gets the missing initial data from the video server and the future data from the live channel. We now discuss the data transmission method used by the video server.

a) Data Transmission Method

All video segments S_i ($i = 1,2,\cdots,K$) are of uniform size (in time units) and they are constructed as discussed above. The video server broadcasts the segments as follows:

1) The first channel C_1 broadcasts the first segment S_1, repeatedly.

2) The second channel C_2 transmits next two segments S_2 and S_3, alternately and repeatedly.

3) The ith channel C_i ($i > 2$) broadcasts $3*2^{i-3}$ number of segments from ($3*2^{i-3} + 1$) to ($3*2^{i-2}$), sequentially and periodically.

Let the video server allocate N video channels C_1, $C_2,\cdots,C_N$ to the desired video. After downloading the first segment S_1, the video server broadcasts this segment over its first video channel C_1, repeatedly, and concurrently downloads the second segment S_2 into its buffer from the live system. After the video server stores the segment S_2 into buffer from the live system, it broadcasts S_2 from the next time slot along with S_1 according to the data transmission Method. When the video server broadcasts S_1 and S_2, it stores third segment S_3 from the live system into its buffer and then broadcasts S_3 along with S_1 and S_2. A user request is allowed to get video data from the live system or the video server at the starting point of a time slot, not in between; thus, a user may have to wait for at most one time slot. If a user request is received when the live system broadcasts S_1, it would get the data from the live system at the start of the second time slot, but by that time this request must has missed S_1. The segment S_1 has already been stored by the video server in its buffer in the first time slot and in the next time slot, *i.e.*, second time slot, it is broadcast by the

video server as per the data transmission method. Thus, the request can get S_1 from the video server and its storage requirement is equal to a segment size. The video server downloads future data from the live channel into its buffer and broadcasts the already stored video data from its buffer, if there is a free video channel available. If the live video broadcasting is not over and all video channels of the video server have been exhausted, then there is need to make a video channel free to broadcast the newly stored video segments. The possible solution to handle this problem is to make the last video channel free by transferring its data to other video channels. The important issue while transferring the data from one video channel to other is that the requests which are currently viewing and those which would view in future should get continuous delivery of the video data. For transferring data from the last video channel to other video channels, we need to increase the segments' size. The data transferring approach without disturbing user services is called channel transition mechanism. The channel transition can be an intermediate in which the total size of the video is still unknown, or it can be the final channel transition when the video size is known, *i.e.*, the live video transmission is over.

b) Intermediate Channel Transition

The important point in a channel transition is that the users who have been viewing since prior to the channel transition and those who would view after the channel transition should get continuous delivery of the video data. Here we discuss a channel transition when all video channels allocated to the video by the video server have been exhausted and the live video is still going on. After carrying out the channel transition, the size of a (new) segment becomes double of that of an old segment. Denote old segments before the ith channel transition by S_k^{i-1}, S_{k+1}^{i-1}, ···. and new segments after transition by S_k^i, S_{k+1}^i, ··· Then, $S_k^i = S_{2k-1}^{i-1} + S_{2k}^{i-1}$, ···. After the channel transition, the waiting time for a user request would be equal to two segments. Therefore, it is necessary to delay the channel transition as much as possible, while maintaining continuous delivery of the video data to users. Continuous delivery can be ensured if the channel transition takes place when the second segment S_2 has been transmitted over the second channel C_2. If the channel transition is made after the third segment S_3 has been transmitted over C_2, then the requests that start receiving the video data from the time slot just before the channel transition will not get the data in time because half of the new second segment S_2^1 (which is the old segment S_3^0, S_3^0 is the original third segment S_3) has already been transmitted over C_2 just before the channel transition and S_2^1 will be transmitted over C_2 just after the channel transition. It means that the old users who received the segment S_3 would be expecting S_4. But since the new second segment S_2^1 is transmitted just after the channel transition and its first half, *i.e.*, S_3 will be received again, not S_4. Thus, all users will not get the required data in time. We considered the second channel as an example, but similar problems occur with other channels, too. Non-delivery of the video data can be overcome if the channel transition is made when the second segment S_2 has been transmitted over the second channel C_2. We now illustrate the channel transition with an example.

Illustration

Assume that the video server allocates five video channels to the video. The number of segments that can be transmitted over these five channels is $3*2^{N-2} = 3*2^{5-2} = 24$. Let the live video start at time t_0. The video server is always tuned to the live channel to store the video data into its buffer from the live system. Let the size of a time slot, which is also equal to a segment size (in time units), be 1.0 minute. The video server first downloads video data from the live system for 1.0 minute into its buffer, denoting it as S_1, and then broadcasts this data as per the data transmission method. The requests which have arrived by the time t_0 get video data from the live system. The requests which would arrive after the live video has been started (say, at time $t_0 + 0.5$) would get video data from the live system at the start of the next time slot, *i.e.*, at time $(t_0 + 1.0)$. Call time durations from t_0 to $(t_0 + 1.0)$, $(t_0 + 1.0)$ to $(t_0 + 2.0)$,···, $(t_0 + \text{i})$ to $(t_0 + (\text{i} + 1))$, ··· as 0th time slot T_0, 1st time slot T_1, 2nd time slot T_2, ···, *i*th time slot T_i, ···, respectively. Denote the data broadcast by the live system in these time slots by S_{0L}, S_{1L}, $S_{2L,}$ ···,$S_{iL,}$ ···. We denote the video data available in buffer of the video server for broadcasting in the time slots T_0, T_1, T_2, ···, T_i, ···, respectively, by segments S_0, S_1, $S_{2,}$ ···,$S_{i,}$ ···. It may be seen that $S_0 = 0$, $S_1 = S_{0L}$, $S_2 = S_{1L}$, ···. The segment S_0 is zero because no data is available in buffer of the video server for broadcasting in the time slot T_0. *The segment stored into buffer in current time slot will be available for broadcasting in next time slot*, *not in the current time slot.* The video server stores S_1 into its buffer in time slot T_0 from the live system and this will be available for broadcasting in time slot T_1. It is to note that the request, R_0, arrived at any time in 0th time slot T_0 will not get S_1 from the live system because R_0 would be allowed to receive data from the live system from the time $t_0 + 1.0$ onward and by that time the live system would have already broadcast S_1. However, the video server has stored S_1 into its buffer in 0th time slot T_0 and broadcasts it from 1st time slot as per the data transmission method. The request R_0 can get S_1 from the video server and the future data from the live system. It is noteworthy to mention that the live system and the video server broadcast the

video data, so any number of requests received in any time slot will require same amount of resources as a single request. Therefore, without loss of generality we can represent all the requests received in a time slot by a single request. The requests received in the *i*th time slot are denoted by R_i. Consider request R_2 that arrives in 2nd time slot T_2. This request will be allowed to join the live channel at time $t_0 + 2.0$ for receiving the future data. So, R_2 does not get S_1 and S_2 because their transmission has already been over by the live system. However, the video server has stored S_1 and S_2 in its buffer in the time slots T_0 and T_1, respectively, from the live system and broadcasts S_1 from time slot T_1 onward and S_2 from time slot T_2 onward. Thus, R_2 can get S_1 and S_2 from the video server. Using similar argument, it is not difficult to show that a request received in any time slot would get the required data in time. This process will continue till all time slots of all video channels have been occupied and the live broadcasting is still there. When all video channels have been exhausted, we need to perform the channel transition to make the last channel free for broadcasting the new segments. It means that the segments occupied by the 5th channel C_5 (*i.e.*, S_{13}, S_{14}, ⋯, S_{24}) need be broadcast using the first four channels. In the modified conservative scheme, it can easily be done by just making the segment size double because the video channel C_k ($k > 2$) can occupy maximum number of segments that is equal to the sum of all segments occupied by all lower indexed video channels C_{k-1}, C_{k-2}, ⋯, C_1. Denote new segments and new time slots by S_0^1, S_1^1, S_2^1, ⋯, S_i^1, ⋯, and T_0^1, T_1^1, T_2^1, ⋯, T_i^1, ⋯, respectively. The size of a new segment (or time slot) is double of that of an old segment (or time slot). Here T_0^1 denotes the time slot just prior to the channel transition and S_0^1 denotes the data downloaded in buffer in time slot T_0^1. The segment S_0^1 contains data of segments that have been stored in the buffer in the time slot just before the channel transition. **Figure 2** shows the first channel transition at thick line of the time point for using five video channels. In **Figures 2-5**, the gray-colored channels represent the live channels and the others are video channels allocated to the video by the video sever. The optimal time point at which the channel transition should be made is ($t_0 + 24$), *i.e.*, at the end of the time slot T_{24} because by that time all time slots of all the video channels of the video server must have been occupied. After carrying out the channel transition, the first new segment S_1^1 comprises S_1 and S_2. The second and third new segments (S_2^1 and S_3^1) comprise S_3 & S_4 and S_5 & S_6 segments, respectively. The transmission of new segments over the video channels takes place exactly in the same way as the old segments according to the data transmission method. The important characteristics of the conservative staircase scheme is that for freeing the last video channel all segments are made double and the channel transition can be delayed optimally. The *i*th new segment and the *i*th new time slot can be written in terms of old segments and old time slots, respectively, as

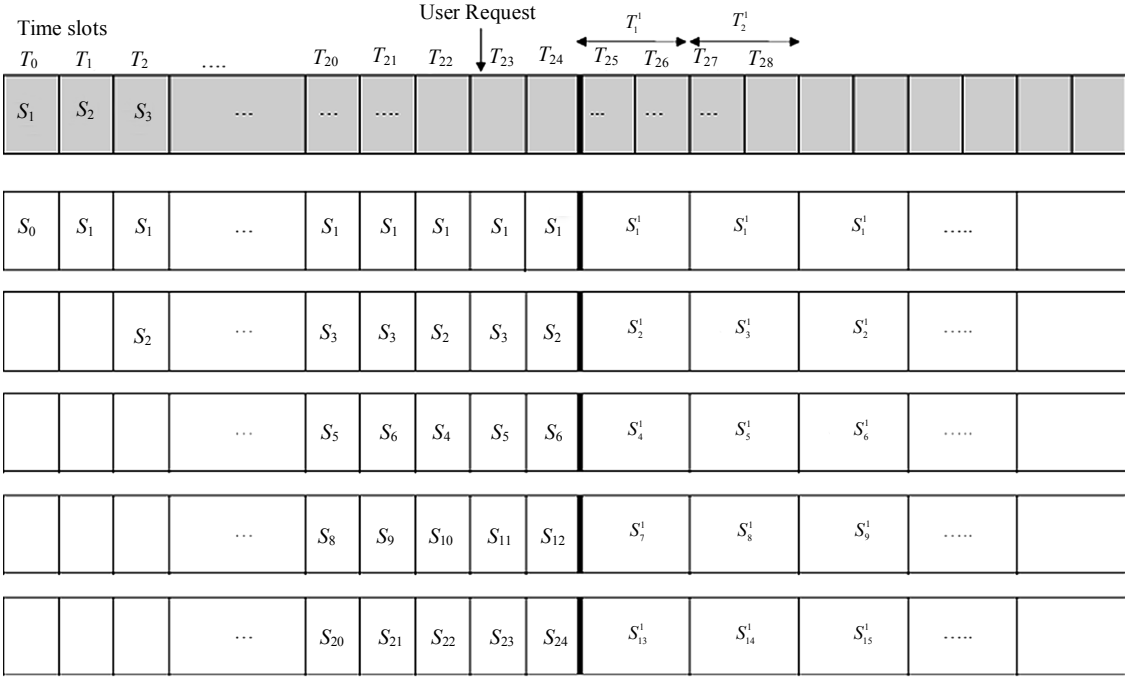

Figure 2. First channel transition.

$$S_i^1 = S_{2i-1}S_{2i}$$

and

$$T_i^1 = T_{24+2i-1} + T_{24+2i};$$

For i = 1,2, ···, we have

$$S_1^1 = S_1S_2, \quad S_2^1 = S_3S_4, \quad S_3^1 = S_5S_6, \cdots,$$
$$T_1^1 = T_{25} + T_{26}, \; T_2^1 = T_{27} + T_{28}, \; T_3^1 = T_{29} + T_{30}, \cdots,$$

Consider request R_{23} received at any time in 23rd time slot T_{23} (refer to **Figure 2**). This request gets S_1 from the channel C_1, S_2 from the 2nd channel C_2, S_6 from the 3rd channel C_3, S_{12} from the 4th channel C_4, in the time slot T_{24}, and the segments S_{24} onward from the live system. The request R_{23} would require S_{24} for viewing in the T_{47} time slot in terms of new segments. The remaining data (*i.e.*, segments S_1 to S_{23}) is provided by the video server (refer to **Figure 2**). The segment S_3, first part of the segment S_2^1 is provided by the video server just after the channel transition followed by S_4 as it is the second half of S_2^1. The request received after the channel transition gets video data uninterruptedly in terms of new segments. In fact, we can show that for any request received after or before the channel transition will always get the required data in time. This process continues till all new time slots of all video channels have been occupied. Since the size of a current segment is twice of that of an old one, the next time channel transition will take place when there are 24 new segments or 48 old segments. So far the video has been played for 24 minutes. Next time the channel transition will take place when the video must have been played for 48 segments, *i.e.*, 48 minutes. This process will continue for the duration of the live video transmission. **Figure 3** shows the second channel transition. The user's waiting time after the second transition will be 4 minutes as the segment size is four times that of the original one.

c) Final Channel Transition

We now discuss the final channel transition, which is performed only after the live video has been over. To carry out the final channel transition, the number of segments on the last video channel must be less than its capacity. If the number of segments transmitted by the last channel is equal to its capacity, we do nothing and this is the best scenario. Here the "capacity" means the maximum number of segments that can be transmitted by that channel. The final channel transition is necessary for utilizing bandwidth efficiently. Here our objective is that the video segments should occupy all time slots on all video channels. We may assume that the video data always comprises integral segments. If the last segment is not a complete, then this is made a complete segment by adding dummy data. The channel C_1 transmits the segment S_1^{L-1} and the channel C_2 transmits the segments S_2^{L-1} & S_3^{L-1} in normal course of time. After the final channel transition, the segment size decreases as the

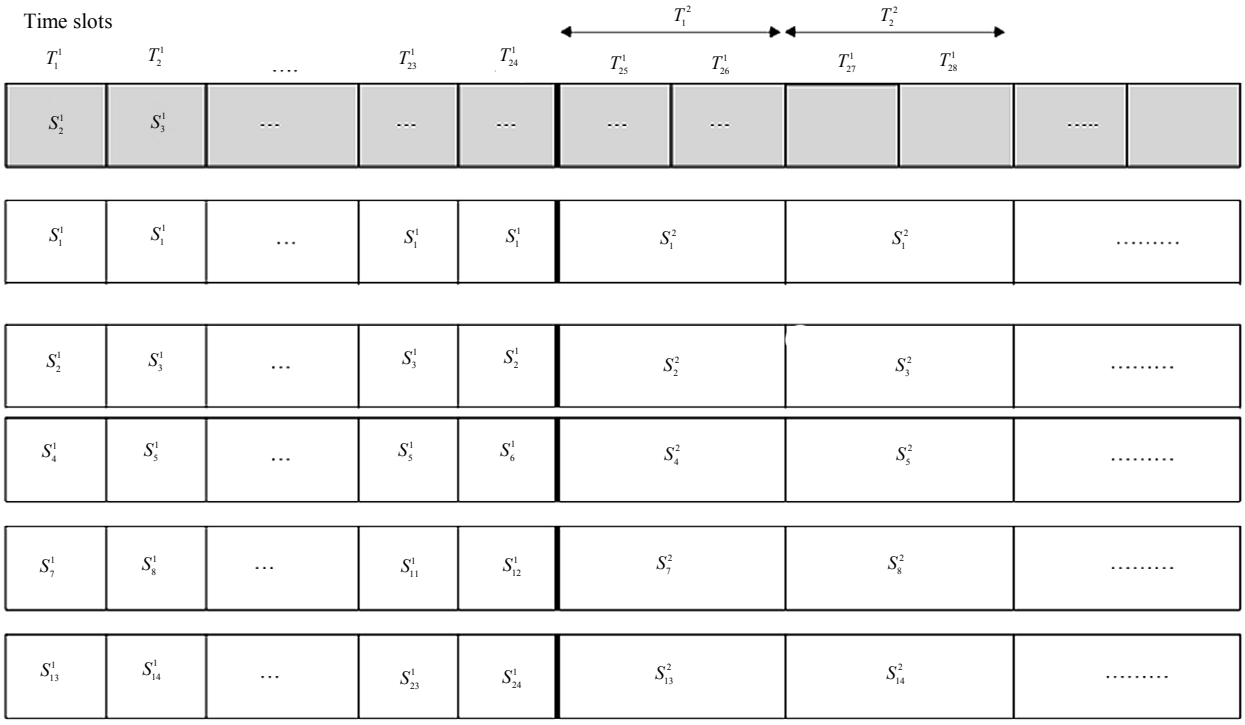

Figure 3. Second channel transition.

number of segments increases. In other words, some last portion of the segment S_1^{L-1} is added to the beginning of S_2^{L-1} and some last portion S_2^{L-1} is added to the beginning of S_3^{L-1}, and so on. In this way, we increase the number of segments. By doing so, these new segments will occupy all time slots of all video channels. Here an important question is "*will all users get video data in time*?" If not, how to make the segments' allocation over the video channels so that all users can get continuous delivery of the video data. We illustrate this with an example. Let the video be allocated five video channels by the video server. The last channel, fifth one, can occupy 12 segments (from 13th to 24th). The live video can be over at any time, *i.e.*, after 12th or 13th, ⋯ or 24th segment. If the live video is over after the 24th segment, we do nothing. Assume that the live video is over in the T_i^{L-1} ($12 < i < 24$) time slot in which the ith segment S_i^{L-1} has been downloaded. We would need to carry out the last channel transition after T_i^{L-1} time slot. By delaying one time slot, we get one time slot free on the last video channel and that time slot is used to broadcast the segment S_2^{L-1} or S_3^{L-1} just before the final channel transition depending upon whether the last video segment broadcast by the live channel was even or odd. This is shown as gray-colored time slot on the last channel in **Figure 4**.

Consider request R_{12} that begins downloading video data into its buffer from the live system from the time slot T_{13}^{L-1} onward. It can download, if required, the segments S_4^{L-1}, S_7^{L-1}, S_{13}^{L-1} in T_{13}^{L-1} time slot and the segments S_2^{L-1}, S_5^{L-1}, S_8^{L-1}, S_3^{L-1} in time slot T_{14}^{L-1} from the 2nd, 3rd, 4th, and 5th video channels, respectively. The segment S_1^{L-1} can be viewed while downloading from the first video channel and does not require any storage. The request R_{12} has all initial segments except the segment S_6^{L-1}. This segment would be required for viewing after the segment S_5^{L-1}. After the channel transition, the segment S_6^{L-1} is distributed among the segments S_{10}^{L}, S_{11}^{L}, & S_{12}^{L}. These segments can be downloaded in time while the segments S_2^{L-1}, S_3^{L-1}, S_4^{L-1} are viewed. Consider another request R_{13} that begins downloading the video data into its buffer from the video server from the time slot T_{14}^{L-1} onward and can store, if required, the segments S_2^{L-1}, S_5^{L-1}, S_8^{L-1}, S_3^{L-1} into its buffer. The segment S_4^{L-1} will be required for viewing after the segment S_3^{L-1}. But, this segment is neither in buffer nor available for downloading and after the channel transition will be distributed among the segments S_6^{L}, S_7^{L}, and S_8^{L}. These segments can be downloaded in time while the segments S_2^{L-1} and S_3^{L-1} are viewed because the data required for two old time slots (before the channel transition) would be sufficient for viewing in more

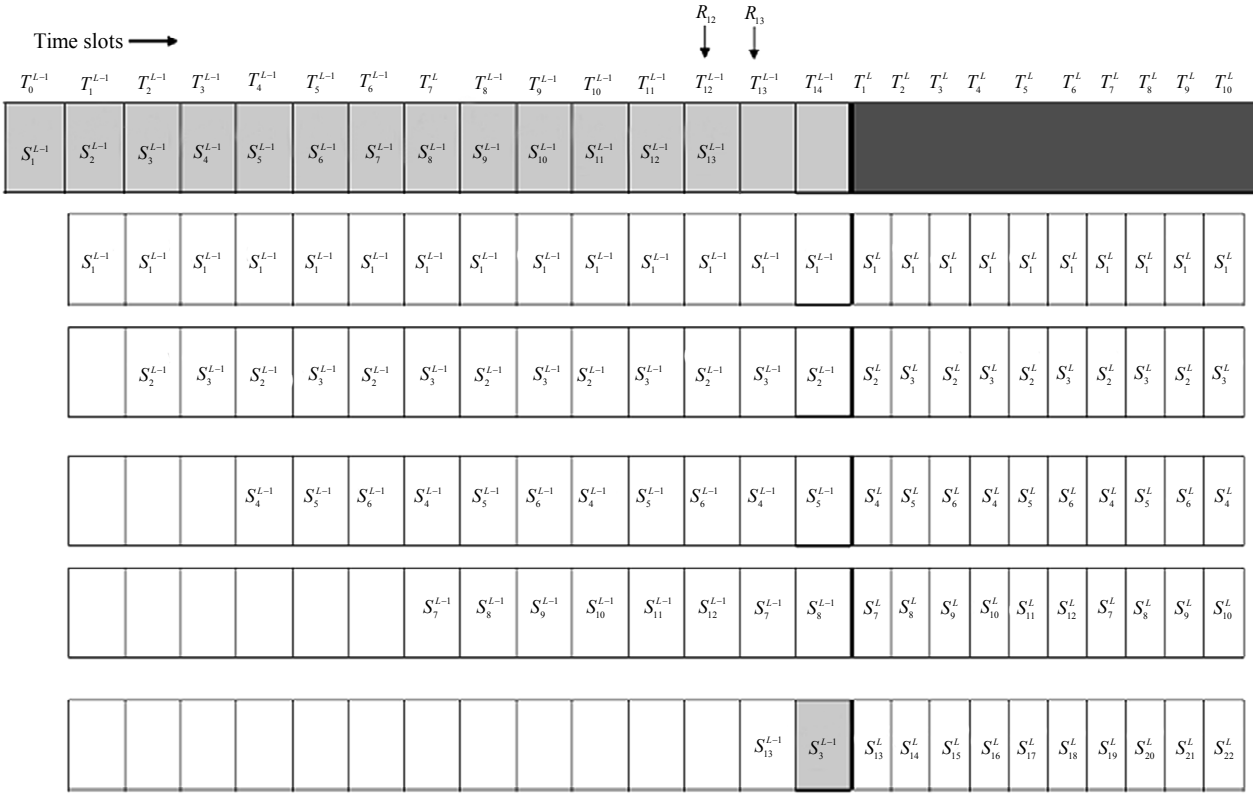

Figure 4. Final channel transition.

than three new time slots (after the channel transition). If any problem related to the data availability is there, it will be for the S_4^{L-1} segment. The request which may have problem of data availability is one that starts receiving video data from the time slot just before the channel transition. For other requests whether received before or after the channel transition, there is no problem of data availability. **Figure 4** shows the final channel transition when the live video is over just after the live system has broadcast the segment S_{13}^{L-1} in T_{13}^{L-1} time slot. Consider another case when the live video is over after the segment S_{14}^{L-1} has been broadcast by the live system. We need to perform channel transition after the time slot T_{15}^{L-1} (it is not shown in figure because of size). The request R_{14} can download, if required, the segments S_3^{L-1}, S_6^{L-1}, S_9^{L-1}, S_2^{L-1} in time T_{15}^{L-1} time slot before the channel transition. It however does not have the segment S_4^{L-1}, which is a part of the segments S_6^L and S_7^L. These segments can be downloaded into buffer in time while the segments S_2^{L-1} and S_3^{L-1} are viewed because the duration of these two segments (*i.e.*, S_2^{L-1} & S_3^{L-1}) is more than that of the three new segments (after the channel transition). So, the segments S_6^L and S_7^L can be downloaded in time. Consider another request, say $R_{21,}$ which receives video data from the video server from the time slot T_{22}^{L-1} onward. In T_{22}^{L-1} time slot, the segments S_2^{L-1}, S_5^{L-1}, S_8^{L-1}, S_3^{L-1} can be downloaded, if required, but the segment S_4^{L-1} is not in buffer and after the channel transition this segment gets distributed among the segments S_4^L and S_5^L. These segments can be downloaded in time when the segments S_2^{L-1} & S_3^{L-1} are viewed. Now the only point to resolve is "*what segments after the channel transition are into which the segment* S_4^{L-1} *is distributed.*" The smallest and largest indices of new segments (after the channel transition), denoted by I_S and I_L, which contain the data of segment S_4^{L-1} are given by

$I_S = n$ such that $\min_n \left\lfloor \frac{n*p}{K} \right\rfloor \geq 3$ and $I_L = n$ such that $\min_n \left\lfloor \frac{n*p}{K} \right\rfloor \geq 4$

where p is the index of the last segment broadcast by the live system and K is the number of video segments.

For example, consider that the last segment broadcast by the live system is S_{16}^{L-1}. The request R_{16} receives video data from the video server in T_{17}^{L-1} time slot onward, the segment S_4^{L-1} would be distributed among the segments S_5^L and S_6^L. This can easily be verified as follows:

$$S_1^L = \frac{16}{24} S_1^{L-1}; \quad S_2^L = \frac{8}{24}\left(S_1^{L-1} + S_2^{L-1}\right); \quad S_3^L = \frac{16}{24} S_2^{L-1};$$

$$S_4^L = \frac{16}{24} S_2^{L-1}; \quad S_5^L = \frac{8}{24}\left(S_3^{L-1} + S_4^{L-1}\right); \quad S_6^L = \frac{16}{24} S_4^{L-1}.$$

4. Results and Discussion

The conservative scheme provides video data to users in time, whereas the staircase scheme does not. That is the reason we have considered the conservative scheme for live video broadcasting. Another important characteristics of this scheme is that the number of segments occupied by a video channel C_i is the sum of all the segments transmitted by all video channels having indices 1, 2, ···, $i - 1$. Because of this the channel transition can be done at optimal time point, *i.e.*, the transition can be delayed till all time slots of all video channels have been occupied. The buffer storage requirement depends upon the arrival time of the request, but in no case it can be more than 50% of the video length. Consider **Figure 5** in which the gray-colored channel is the live channel and the dark black line in each channel is the channel transition point.

Using similar discussions, we can find out the buffer requirement for any request. **Table 1** shows the buffer requirements for different requests (referring to **Figure 5**) for allocating five video channels to the video.

In **Table 2**, for R_{12} and R_{13} requests, there are two different storage requirements. If the live video is over after the 24th segment, then it is 11S and 11S, respectively; otherwise 12S and 13S. The waiting time in this scheme is pre-decided for the initial users and remains same till the channel transition time. After every channel transition except the last one the waiting time becomes double. When the live transmission is over, the final channel transition is carried out and then the user's waiting time is stabilized. We can find out how many and what the initial time slots are in a new time slot after the live video is over. The size of a time slot after the channel transition except the last one becomes double. If T_i and T_i^1 are the ith time slots before and after the first channel transition, then we have the following relation:

$$T_i^1 = T_{24+2i-1} + T_{24+2i}$$

In general, we have

$$T_i^k = T_{24+2i-1}^{k-1} + T_{24+2i}^{k-1}, \quad \text{for} \quad k = 1, 2, \cdots, L-1, \quad (1)$$

where T_i^0 denotes the very first ith time slot, *i.e.*, $T_i^0 = T_i$ and L specifies the final channel transition.

It is to note that till the final but one channel transition the time slots become double of the previous ones. We can find the size of a time slot after any channel transition in terms of the original time slots. For example, consider fourth channel transition (assuming it is not the last channel transition). Then, from (1), we have

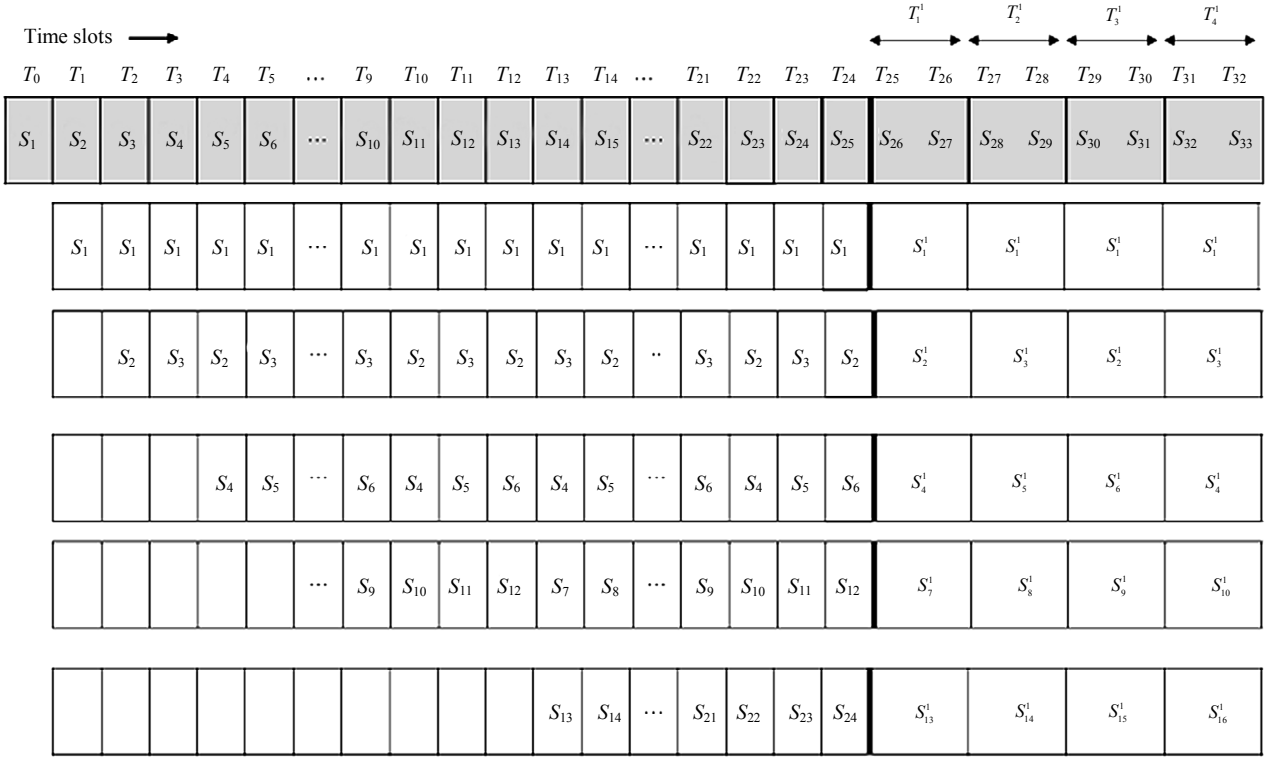

Figure 5. Availability of segments over different channels before and after the channel transition.

Table 1. Segments (seg.) stored from the live system and video server (VS) for R_{10}.

Time slot	*Seg. available for storing from VS*	*Seg. stored from VS*	*Seg. stored from live system*	*Seg. required for viewing*	*Total seg. required*
T_{11}	$S_2 + S_6 + S_{10}$	S_2	S_{12}	S_1	+1 + 1 = 2
T_{12}	$S_3 + S_4 + S_{11}$	$S_3 + S_4$	S_{13}	S_2	+2 − 1 + 1 = 2
T_{13}	S_5	S_5	S_{14}	S_3	+1 − 1 + 1 = 1
T_{14}	S_6	S_6	S_{15}	S_4	+1 − 1 + 1 = 1
T_{15}	S_7	S_7	S_{16}	S_5	+1 − 1 + 1 = 1
T_{16}	S_8	S_8	S_{17}	S_6	+1 − 1 + 1 = 1
T_{17}	S_9	S_9	S_{18}	S_7	+1 − 1 + 1 = 1
T_{18}	S_{10}	S_{10}	S_{19}	S_8	+1 − 1 + 1 = 1
T_{19}	S_{11}	S_{11}	S_{20}	S_9	+1 − 1 + 1 = 1
T_{20}			S_{21}	S_{10}	−1 + 1 = 0
T_{21}			S_{22}	S_{11}	−1 + 1 = 0
		Buffer Storage required for request R_{10} = 11S			

In last column "+" and "–" sign indicate that segment is stored in buffer and read from buffer., e.g., +1 − 1 + 1 = 1 means 1 segment are stored from the video server, 1 segment is read from buffer, and 1 segment is stored from the live channel into buffer. Thus, net requirement is 1 segment. S_1 is viewed while down-loading.

Table 2. Buffer requirement for different requests allocating five video channels.

Request	*Buffer Requirement*	*Request*	*Buffer Requirement*
R_0	S	R_7	8S
R_1	2S	R_8	9S
R_2	3S	R_9	10S
R_3	4S	R_{10}	11S
R_4	5S	R_{11}	12S
R_5	6S	R_{12}	11S or 12S
R_6	7S	R_{13}	11S or 13S

$$T_i^4 = T_{24+2i-1}^3 + T_{24+2i}^3 \quad (2)$$

We can take $i = 1$ because after any channel transition all time slots are of same durations. Thus, we have from (2),

$$T_1^4 = T_{25}^3 + T_{26}^3$$

We now need T_{25}^3 and T_{26}^3, which are given by

$$T_{25}^3 = T_{24+49}^2 + T_{24+50}^2 = T_{73}^2 + T_{74}^2;$$
$$T_{26}^3 = T_{24+51}^2 + T_{24+52}^2 = T_{75}^2 + T_{76}^2.$$

Again T_{73}^2, T_{74}^2, T_{75}^2, and T_{76}^2 are needed and they are given by

$$T_{73}^2 = T_{24+145}^1 + T_{24+146}^1 = T_{169}^1 + T_{170}^1$$
$$T_{74}^2 = T_{24+147}^1 + T_{24+148}^1 = T_{171}^1 + T_{172}^1$$
$$T_{75}^2 = T_{24+149}^1 + T_{24+150}^1 = T_{173}^1 + T_{174}^1$$
$$T_{76}^2 = T_{24+151}^1 + T_{24+152}^1 = T_{175}^1 + T_{176}^1$$

We need T_{169}^1, T_{170}^1, T_{171}^1, T_{172}^1, T_{173}^1, T_{174}^1, T_{175}^1, and T_{176}^1, and they are given by

$$T_{169}^1 = T_{24+337}^0 + T_{24+338}^0 = T_{361}^0 + T_{362}^0$$
$$T_{170}^1 = T_{24+339}^0 + T_{24+340}^0 = T_{363}^0 + T_{364}^0$$
$$T_{171}^1 = T_{24+341}^0 + T_{24+342}^0 = T_{365}^0 + T_{366}^0$$
$$T_{172}^1 = T_{24+343}^0 + T_{24+344}^0 = T_{367}^0 + T_{368}^0$$
$$T_{173}^1 = T_{24+345}^0 + T_{24+346}^0 = T_{369}^0 + T_{370}^0$$
$$T_{174}^1 = T_{24+347}^0 + T_{24+348}^0 = T_{371}^0 + T_{372}^0$$
$$T_{175}^1 = T_{24+349}^0 + T_{24+350}^0 = T_{373}^0 + T_{374}^0$$
$$T_{176}^1 = T_{24+351}^0 + T_{24+352}^0 = T_{375}^0 + T_{376}^0$$

Here $T_i^0 s$ denote original time slots. So, we have

$$T_1^4 = T_{361} + T_{362} + \cdots + T_{376}$$

The time slot after the final channel transition (*i.e.*, Lth) is α times of a time slot of that of $(L-1)$th channel transition, where $0.5 \leq \alpha \leq 1$. The value of $\alpha = 0.5$ means that the live video was over at the time when all time slots of all video channels had been occupied by the segments. In that case the last channel transition was not required. This situation is exactly same for $\alpha = 1$. If $\alpha = 1$, then the live video transmission is over just after all time slots of all the channels have been occupied. In this case, the user's waiting time is unchanged. It means that after the final but one channel transition, the number of segments is such that all time slots of the last channel have been occupied and we need not do anything. Here we have discussed for values of $\alpha = 0.5$ and 1. The exact value of α for other cases will depend on when the live video is over. Since we do not know in advance when the live video will be over, the exact value of α cannot be determined in advance. When the live video is over, the entire video data is distributed on all channels as per the scheme's basic architecture.

In other broadcasting schemes including the conservative staircase scheme the user waiting time is decided by the bandwidth allocated to the video, *i.e.*, the size of a video segment, whereas in the proposed scheme it varies after every channel transition. The initial waiting time is decided by the service provider. As we know that the segment size becomes double of the previous size after every channel transition except the last channel transition, so is the user's waiting time. As far as performance of the proposed scheme is concerned, there does not seem to appear alternative work in literature to make a meaningful comparison and hence the comparison is not possible.

5. Conclusions

In this paper, we have proposed a technique for supporting the live video in the conservative scheme. The important characteristics of the conservative scheme is that the number of segments transmitted by a video channel is equal to the sum of all segments transmitted by all lower-indexed channels. This characteristic has been exploited to develop a mechanism for the live video. Providing live video services has wide applications, such as cricket match, interactive education session, etc.

6. References

[1] Y.-C. Tseng, M.-H. Yang and C.-H. Chang, "A Recursive Frequency-Splitting Scheme for Broadcasting Hot Videos in VOD Service," *IEEE Transactions on Communications*, Vol. 50, No. 8, 2002, pp. 1348-1355. doi:10.1109/TCOMM.2002.801466

[2] W. F. Poona, K.-T. Lo and J. Feng, "First Segment Partition for Video-on-Demand Broadcasting Protocols," *Computer Communications*, Vol. 26, No. 14, 2003, pp. 1698-1708. doi:10.1016/S0140-3664(02)00264-5

[3] S. Vishwanathan and T. Imielinski, "Metropolitan Area Video on Demand Service Using Pyramid Broadcasting," *Multimedia Systems*, Vol. 4, No. 4, 1996, pp. 197-208. doi:10.1007/s005300050023

[4] L. Gao, J. Kurose and D. Towsley, "Efficient Schemes for Broadcasting Popular Videos," *Multimedia Systems*, Vol. 8, No. 4, 2002, pp. 284-294. doi:10.1007/s005300100049

[5] Ch.-L. Chan, S.-Y. Huang, T.-C. Su and J.-S. Wang, "Buffer-Assisted on-Demand Multicast for VOD Applications," *Multimedia Systems*, Vol. 12, No. 2, 2006, pp. 89-100. doi:10.1007/s00530-006-0041-1

[6] S. Ramesh, I. Rhee and K. Guo, "Multicast with Cache (Mcache): An Adaptive Zero-Delay Video-on-Demand Service," *IEEE Proceedings of* 2*th Annual Joint Confer-*

ence of the Computer and Communications Societies, Vol. 1, Anchorage, 22-26 April 2001, pp. 85-94.

[7] L. S. Juhn and L.-M. Tseng, "Harmonic Broadcasting Scheme for Video-on-Demand Service," *IEEE Transactions on Consumer Electronics*, Vol. 43, No. 3, 1997, pp. 268-271.

[8] J.-F. Paris, S. W. Carter and D. D. E. Long, "Efficient Broadcasting Protocols for Video on Demand," *Proceedings of 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Montreal, 19-24 July 1998, pp. 127-132.

[9] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video on Demand Systems," *ACM SIGCOMM Computer Communication Review*, Vol. 27, No. 4, 1997, pp. 89-100. doi:10.1145/263109.263144

[10] Z. Q. Gu and S. Y. Yu, "Conservative Staircase Data Broadcasting Protocol for Video on Demand," *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 4, 2003, pp. 1073-1077. doi:10.1109/TCE.2003.1261198

[11] L. S. Juhn and L. M. Tseng, "Fast Data Broadcasting and Receiving for Popular Video Service," *IEEE Transactions on Broadcasting*, Vol. 44, No. 1, 1998, pp. 100-105. doi:10.1109/11.713059

[12] L. S. Juhn and L. M. Tseng, "Staircase Data Broadcasting and Receiving Scheme for Hot Video Service," *IEEE Transactions on Consumer Electronics*, Vol. 43, No. 4, 1997, pp. 1110-1117. doi:10.1109/30.642378

[13] S. Chand, "Live Video Services Using Fast Broadcasting Scheme," *Journal of Communications and Network*, Vol. 2, No. 1, 2010, pp. 79-85. doi:10.4236/cn.2010.21013

