Research Article

Single-Channel Data Broadcasting under Small Waiting Latency

Hsiang-Fu Yu

Department of Computer Science, National Taipei University of Education, Taipei 10671, Taiwan

Correspondence should be addressed to Hsiang-Fu Yu; yu@tea.ntue.edu.tw

Received 5 March 2013; Accepted 18 March 2013

Academic Editor: Pin-Han Ho

Copyright © 2013 Hsiang-Fu Yu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Due to the advancement of network technology, video-on-demand (VoD) services are growing in popularity. However, individual stream allocation for client requests easily causes a VoD system overload; when its network and disk bandwidth cannot match client growth. This study thus presents a fundamentally different approach by focusing solely on a class of applications identified as latency tolerant applications. Because video broadcasting does not provide interactive (i.e., VCR) functions, a client is able to tolerate playback latency from a video server. One efficient broadcasting method is periodic broadcasting, which divides a video into smaller segments and broadcasts these segments periodically on multiple channels. However, numerous practical systems, such as digital video broadcasting-handheld (DVB-H), do not allow clients to download video data from multiple channels because clients usually only have one tuner. To resolve this problem in multiple-channel broadcasting, this study proposes a novel single-channel broadcasting scheme, which leverages segment-broadcasting capability further for more efficient video delivery. The comparison results show that, with the same settings of broadcasting bandwidth, the proposed scheme outperforms the alternative broadcasting scheme, the hopping insertion scheme, SingBroad, PAS, and the reverse-order scheduling scheme for the maximal waiting time.

1. Introduction

Due to the advancement of network technology, video-on-demand (VoD) services are growing in popularity. Clients can watch their desired videos at anytime without waiting or visiting a video rental store. Because of the online access provided by VoD services, several studies have predicted the success of VoD [1, 2]. VoD is inherently a personalized service because of its characteristic one-to-one interaction. Therefore, a VoD system typically allocates a dedicated stream for each incoming video request [3]. However, individual stream allocation easily causes a VoD system overload when its network and disk bandwidth cannot match client growth. This study thus presents a fundamentally different approach by focusing solely on a class of applications identified as latency tolerant applications. The key feature of latency tolerant applications is that they are unconcerned with latency between video servers and clients. Broadcast video streaming is perhaps the most important example of this class of applications [4]. Because video broadcasting does not provide interactive (i.e., VCR) functions, a client is able to tolerate playback latency from a video server. One efficient broadcasting method is periodic broadcasting, which divides a video into smaller segments and broadcasts them periodically on a set of communication channels. This method enhances bandwidth usage by allowing various clients to share the same channel bandwidths. Because periodic broadcasting typically requires a client to wait for the beginning of the first segment before starting playback, this scheme cannot support real-time VoD services.

The fast broadcasting (FB) [5] scheme divides a video into a geometrical series of 1, 2, 4, \ldots, 2^{k-1}, where \( k \) is the number of broadcasting channels. An implementation of the FB scheme on IP multicasting was reported in [6]. To achieve minimal latency, the harmonic broadcasting (HB) scheme [7] partitions a video into multiple segments, and each segment \( S_i \) is divided into \( i \) subsegments. The subsegments of the same segment are then broadcast on the same channel. The recursive frequency-splitting (RFS) scheme [8] achieves a near-minimal waiting time by periodically broadcasting each segment at a frequency that can guarantee continuous video
playback. In modifying the FB scheme, the reverse fast broadcasting (RFB) scheme [9] buffers 25% of video length, merely half of what is required by the FB scheme. By combining RFB and RFS, the hybrid broadcasting scheme (HyB) [10] requires the same client buffering space as that of RFB; however, it achieves smaller waiting time. The study in [11] integrates the fixed-delay pagoda broadcasting scheme [12] and RFB to reduce client waiting time and buffer demand. A generalized reverse sequence-based model [13] was proposed to clarify why broadcasting segments in reverse order can reduce buffer requirements. A scalable binomial broadcasting scheme [14] transfers live videos using constant bandwidth, regardless of video length.

The mentioned schemes transfer video segments on multiple channels simultaneously and periodically, and a client typically must receive segments from these channels concurrently. To perform multiple-channel segment broadcasting, a server must multiplex video segments into multiple channels and synchronize these segments across these channels. Segment multiplexing and synchronization are difficult, because packet transmissions are varied with network traffic. In addition, numerous practical systems, such as digital video broadcasting-handheld (DVB-H) and integrated services digital broadcasting-handheld (ISDBH), do not permit a client to download video data from multiple channels, because the client typically has only one tuner [15, 16]. To solve these problems caused by multiple-channel broadcasting, many studies were proposed to broadcast segments over a single channel, such as the alternative broadcasting (AB) scheme [15], the hopping insertion (HI) scheme [16], SingBroad [17], PAS [18], and the reverse-order scheduling (ROS) scheme [19]. The basic concept behind these schemes is to partition a video into equal-sized segments, which are classified into several groups and transferred over a single channel according to a predefined arrangement.

This study proposes a single-channel broadcasting scheme to yield short waiting time. Let $kb$ be the bandwidth of a single channel, where $k$ is a positive integer and $b$ is the playback rate of a video. The proposed scheme partitions the single channel as an infinite set of time slots. Each time slot is further composed of smaller subslots. A video of length $L$ is equally divided into $2^k - 1$ segments, which are then arranged to $k$ groups, denoted by $G_0, G_1, \ldots, G_{k-1}$. A segment of group $G_i$ is split into $2^i$ equal subsegments, which are then placed to individual subslots. The mathematical analysis shows that the maximal client waiting time of the scheme is $((k + 1)L/k(2^k - 1))$. This study also verifies the workability of the scheme and compares it with several current approaches. The comparison results show that, with the same settings of broadcasting bandwidth, the proposed scheme outperforms AB, HI, SingBroad, PAS, and ROS for the maximal waiting time. Extensive simulations also indicate that the proposed scheme requires smaller client buffering space than AB and SingBroad for $k > 4$.

The remainder of this study is organized as follows. Section 2 reviews AB, HI, SingBroad, PAS, and ROS. Section 3 introduces the proposed scheme and verifies its accuracy. Section 4 shows the evaluations of the performance of the scheme, and Section 5 makes a brief conclusion.

### Table 1: List of terms used in this study and their respective definitions.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>Video length</td>
</tr>
<tr>
<td>$b$</td>
<td>Video playback rate</td>
</tr>
<tr>
<td>$kb$</td>
<td>Bandwidth of a single channel, where $k$ is a positive integer</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of segments</td>
</tr>
<tr>
<td>$S_i$</td>
<td>$i$th video segment</td>
</tr>
<tr>
<td>$d$</td>
<td>Segment length of $S_i$</td>
</tr>
<tr>
<td>$S_{ij}$</td>
<td>$j$th subsegment of $S_i$</td>
</tr>
<tr>
<td>$t_s$</td>
<td>Client arrival time</td>
</tr>
<tr>
<td>$T_i$</td>
<td>$i$th time slot, $i \geq 0$</td>
</tr>
<tr>
<td>$w$</td>
<td>Maximal waiting time for playback</td>
</tr>
</tbody>
</table>

### 2. Related Work

This section introduces AB [15], HI [16], SingBroad [17], PAS [18], and ROS [19]. Table 1 defines the terms used in this study. As mentioned previously, this study divides a single channel into an infinite set of time slots.

The AB scheme [15] splits a video into $N$ segments. This scheme proposes two modes for determining the value of $N$. One is the mechanism-dominated (MD) mode, and the other is the waiting time-dominated (WD) mode. In the MD mode, $N = \lceil (k + 3)/2 \rceil$, and clients start playing video data when they receive segment $S_1$. The AB scheme with WD obtains $N = \lceil (k + 3)/2 \rceil$. In this mode, the starting time of video playback is determined by whether each segment can be played continuously, rather than the downloading of segment $S_1$. For both modes, the AB scheme broadcasts segment $S_1$ on the single channel at time slot $T_1$ if $i \mod 2 = 0$. The rest of segments are broadcast sequentially and periodically on the remaining time slots. Figure 1 shows an example to demonstrate the segment downloading and playing for AB, where $k = 4$. The segments downloaded and played by a client are gray. The AB scheme with MD divides a video into three segments, as shown in Figure 1(a). A client starts to play video data at the beginning of segment $S_1$. In addition, the AB scheme with WD partitions the same video into four segments, as presented in Figure 1(b). Note that if the client started playing segment $S_1$ on time slot $T_1$, segment $S_2$ would not be played continuously. Therefore, the client begins to play video data on time slot $T_3$ to guarantee continuous playback.

The HI scheme [16] divides a video into $N$ even segments, where $N$ is an arbitrary positive integer. This scheme then classifies the segments into $\lceil N/Q \rceil$ groups, where $Q = \lceil N / \exp(0.57(k - 1)) \rceil$. Group $G_i$ contains segments $S_{jQ+1}$ to $S_{j+1Q}$, where $0 \leq j \leq \lceil N/Q \rceil - 2$. The last group includes the remaining segments. Initially, HI puts the segments of group $G_0$ together in order. The segments of the remaining groups are then inserted into the segments of $G_0$ in a hopping way to obtain the final broadcasting schedule [16].
(3) Partition a single channel as an infinite set of time slots, denoted by $T_0, T_1, T_2$, and so on. Each timeslot is broadcast on time slot $T_k$, where $2 \leq j \leq k - 1$. Let $y$ be zero or a positive integer. Segment of $G_j$ is broadcast on time slot $T_{kj}$. This scheme then puts segments $S_1$ and $S_2$ of $G_j$ on time slots $T_{1+2ky}$ and $T_{1+(k-1)+2ky}$, respectively. The ROS scheme transmits segment $S_{y+2j-1-x}$ of the remaining group $G_j$ on time slot $T_{j+y+x+3x+2y}$, where $2 \leq j \leq k-1$ and $0 \leq x \leq 3\times2^{j-2}-1$. For example, segment $S_6$ of group $G_2$ is broadcast on time slot $T_{2+3y}$ because $j = 2$ and $x = 0$. For $k = 4$, ROS puts segment $S_6$ on time slots $T_2, T_{14}, T_{26}$, and others. When a client wants to watch a video, the client must wait for the beginning of the nearest segment $S_1$ to start downloading. In addition, segments $S_2$ and $S_3$ must be received in order. For the segments of the remaining groups, the client downloads them according to the following process. Suppose that $S_p$ is the segment that a client is currently playing, and segment $S_{p+1}$ of $G_j$ is the segment that appears on the channel and is not received by the client. If $p + 3 \times 2^{i-2} < i$, the client does not download segment $S_i$. Otherwise, the client receives it. When downloading segment $S_1$ is complete, the client starts video playback.

### 3. Proposed Scheme

According to the mentioned schemes [15–19], the number of video segments mainly determines client waiting time. Therefore, the key to minimizing the waiting time is to partition a video into as many segments as possible, under the condition that ensures continuous playback. To maximize the segment number, the proposed scheme broadcasts video data over a single channel according to the following step:

1. Divide a video into $2^k-1$ segments, denoted by $S_1, S_2, \ldots, S_{2^k-1}$ in sequence. The length of each segment, $d$, thus equal $L/(2^k-1)$. For example, in Figure 2, a server allocates a single channel with a bandwidth of $3b$ to broadcast a video of length $L$. The video is equally divided into $3b - 1$ segments, denoted by $S_1, S_2, \ldots, S_7$. The length of each segment equals $L/7$.

2. Classify these segments into $k$ groups, denoted by $G_0, G_1, \ldots, G_{k-1}$. Assemble segments $S_{y+1}$ to $S_{y+2^{k-1}-1}$ into group $G_j$ sequentially. Figure 2 shows that the segments are then classified into three groups $G_0 = \{S_1, G_1 = \{S_2, S_3\}$, and $G_2 = \{S_4, S_5, S_6, S_7\}$. Each segment $S_5$ of group $G_1$ is further partitioned into $2^j$ subsegments, denoted by $S_{5,1}, S_{5,2}, \ldots, S_{5,2^j}$. As shown in Figure 2, segment $S_5$ of group $G_2$ is split into four subsegments $S_{5,1}, S_{5,2}, S_{5,3}$, and $S_{5,4}$. (3) Partition a single channel as an infinite set of time slots, denoted by $T_0, T_1, T_2$, and so on. Each time slot
is used to deliver segment data at a bandwidth of $kb$, and the length of each time slot equals
\[
\frac{L}{(KN)} = \frac{d}{k}. \tag{1}
\]

(4) A time slot $T_i$ is further divided into $2^j$ subslots, denoted by $T_{i1}, T_{i2}, \ldots, T_{i2^j}$, if $i \mod k = j$. The length of a subslot of time slot $T_i$ thus equals $d/(2^j k)$. For example, Figure 2 shows that the length of each time slot equals $d/3$ because $k = 3$. Time slot $T_{i2}$ is further partitioned into four subslots $T_{i2,1}, T_{i2,2}, T_{i2,3}$, and $T_{i2,4}$ because $2 \mod 3 = 2$.

(5) Put the segment data of each group on each time slot in sequence. For example, the segment data of groups $G_0$, $G_1$, and $G_2$ are sequentially broadcast on time slots $T_0$, $T_1$, $T_2$, and so on, as indicated in Figure 2. In general, the segment data of group $G_j$ are put on time slot $T_{j+k^y}$, because there are $k$ groups, where $y$ is zero or a positive integer. Furthermore, the scheme sequentially broadcasts the subsegments of the segments of group $G_j$ on the subslots of time slot $T_{j+k^y}$. For example, Figure 2 shows that the subsegments of segments $S_i$ to $S_j$ of group $G_2$ are sequentially put on the subslots of $T_2, T_3, T_5, T_8, T_{11}$, and others. Note that only a subsegment of a segment of the same group is put on a subslot of a time slot. Because the segment data of group $G_j$ are broadcast once every $k$ time slots and each segment consists of $2^j$ subsegments, each subsegment is transmitted once every $2^j k$ time slots. Therefore, the scheme broadcasts subsegment $S_{ix}$ of group $G_j$ on subslot
\[
T_{j+((x-1)k+2^j y, j-2^j+1)}, \tag{2}
\]
where $y \in \mathbb{Z}$ and $y \geq 0$.

For example, Figure 2 shows that the proposed scheme puts subsegment $S_{13}$ of group $G_2$ on subslot $T_{8+12y,2}$ (e.g., $T_{82}, T_{20,2}$, and $T_{32,2}$) because $k = 3$, $j = 2$, $i = 5$, and $x = 3$.

This study next presents how to download video segments on the client side. A client is assumed to have a sufficient buffer to store downloaded segments. Suppose that a client can download and play the same segment concurrently, because the downloading bandwidth is equal to or larger than the playback rate. This study also assumes that a client desires to watch a video at time $t_a$. Let $T_{u,v}$ be the subslot that is nearest to time $t_a$. The segment downloading and playing are as the following.

(1) The client must wait for subslot $T_{u,v}$ before receiving subsegments. Once the subslot is up, the client starts downloading the subsegment from this subslot.

(2) After this downloading is complete, the client continues to receive the remaining subsegments from the following subslots. If a subsegment has been downloaded, the client simply skips it.

(3) When all the subsegments are received, the client stops the segment downloading.

(4) The client assembles the received subsegments to form complete segments and starts playing them at the beginning of subslot $T_{u+k,v}$.

Figure 3 shows an example for demonstrating how to download and play video segments, where the subsegments downloaded and played by a client are gray. Because subslot $T_{1,2}$ is closest to the client arrival time $t_a$, the client starts downloading subsegment $S_{13}$ on subslot $T_{1,2}$. The client then continues to receive subsegments from subslots $T_{2,3}$ to $T_{3,4}$. Because subsegment $S_{1,1}$ has been downloaded on subslot $T_{3,1,1}$, the client does not receive it again on subslots $T_{6,1}$. Similarly, the client does not receive subsegments $S_{2,1}$ and $S_{2,2}$ on subslots $T_{7,2}, T_{9,1}$, and $T_{10,1}$, and $T_{10,2}$, respectively. When the client finishes receiving all the subsegments at the end of subslot $T_{1,1,4}$, the client stops downloading subsegments. The client assembles the received subsegments to form complete segments and plays them at the start of subslot $T_{4,2}$, as shown in Figure 3.

### 3.1. Workable Verification

Suppose that segment $S_i$ is in group $G_j$, where $2^j \leq i \leq 2^{j+1} - 1$. The mentioned broadcasting process transfers a subsegment of segment $S_i$ once every $k$ time slots. Because the number of subsegments of segment $S_i$ equals $2^j$, the broadcasting process can transmit all the subsegments of segment $S_i$ once every $2^j k$ time slots.
According to the downloading process, a client starts segment downloading at the beginning of subslot \( T_{w,2/k,v} \). Therefore, the client can receive all the subsegments of \( S_i \) at the beginning of subslot \( T_{w,2/k,v} \). In addition, the client begins segment playback at the beginning of subslot \( T_{w,2/k,v} \). Because the playback length of a segment equals \( k \) time slots according to (1), the start time to play segment \( S_i \) is the beginning of subslot \( T_{w,2/k,v} \). To guarantee continuous playback for the client, the end time of downloading segment \( S_i \) must be earlier than the start time of its playback. That is, the beginning of subslot \( T_{w,2/k,v} \) must be later than the beginning of subslot \( T_{w,2/k,v} \). This study evaluates

\[
(u + k + ik) - (u + 2^j/k) = (i + 1 - 2^j)k > 0, \quad \text{due to } 2^j \leq i.
\]

(3)

The end time of downloading segment \( S_i \) is earlier than the start time of its playback. Therefore, the proposed scheme ensures continuous video playback on the client side.

### 4. Performance Analysis and Comparison

This study primarily selected client waiting time and buffer demand as the performance criteria. The proposed scheme was compared with AB, HI, SingBroad, PAS, and ROS. According to the downloading process, when a client exactly arrives at the beginning of a subslot, the waiting time equals \( k \) timeslots (i.e., \( d \)) because of (1). If the client just misses the startup of a subsegment on the channel, the client must additionally wait for the length of the subsegment. Because subsegment \( S_{1,1} \) is the longest subsegment, the maximal waiting time \( w \) equals \( k + 1 \) time slots. That is,

\[
w = \frac{(k + 1) d}{k} = \frac{(k + 1) L}{k(2^k - 1)}.
\]

(4)

Table 2 summarizes the maximal waiting time incurred by AB [15], HI [16], SingBroad [17], PAS [18], ROS [19], and the proposed scheme. The results show that the number of segments mainly determines the maximal waiting times for all the schemes. The increase of the server bandwidth (i.e., the value of \( k \)) enlarges the number of segments and thus reduces the waiting time.

To clarify the performance advantages of the proposed scheme, this study calculated the maximal waiting times of AB, HI, SingBroad, PAS, ROS, and the proposed scheme at various values of \( k \), where the value of \( N \) for HI equals 10000. Figure 4 shows the performance results. As the server bandwidth increases, the waiting times under all the schemes are sharply reduced. In addition, the proposed scheme yields the shortest waiting time. For example, when the server bandwidth equals \( 7b \) (i.e., \( k = 7 \)), the scheme reduces the broadcast latency to less than 0.009L. In contrast, AB-MD, AB-WD, HI, SingBroad, PAS, and ROS yield 0.057, 0.057, 0.019, 0.014, 0.014, and 0.012L, respectively. The proposed scheme reduces the waiting times by 84%, 84%, 53%, 36%, 36%, and 25%. Assume that the video length \( L \) is 120 min. Figure 5 shows the maximal waiting time for all the schemes in seconds. For \( k = 6 \), the waiting times of AB-MD, AB-WD, HI, SingBroad, PAS, ROS, and the proposed scheme are 600,
Table 2: Maximal waiting time for different schemes in terms of video length $L$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>AB-MD</th>
<th>AB-WD</th>
<th>HI</th>
<th>SingBroad</th>
<th>PAS</th>
<th>ROS</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximal waiting time</td>
<td>$\frac{2L}{\lfloor (k+3)/2 \rfloor k}$</td>
<td>$\frac{2L}{\lfloor (k+3)/2 \rfloor k}$</td>
<td>$\frac{2L}{\lfloor (k+3)/2 \rfloor k}$</td>
<td>$\frac{kN}{\exp(0.57(k-1))}$</td>
<td>$\frac{(k-1)L}{(2^{k-1}-1)k}$</td>
<td>$\frac{(k-1)L}{(2^{k-1}-1)k}$</td>
<td>$\frac{(k+1)L}{3 \times 2^{k-3}k}$</td>
</tr>
</tbody>
</table>

where

\[
Q = \left\lfloor \frac{N}{\exp(0.57(k-1))} \right\rfloor
\]

\[
H(i) = \sum_{j=1}^{i} \frac{1}{j}
\]
With low cost and large capacity of storage disks, client buffer demand is no longer a substantial concern. However, for completeness, this work studies the required buffer size under AB-MD, AB-WD, SingBroad, PAS, ROS, and the proposed scheme (the comparison does not include HI, because its buffer requirements are not provided in [16]). Because this study did not derive a close formula for the required buffering space of the proposed scheme, a simulator in Perl [20] was developed to exhaustively search all possibilities to determine the maximum buffering space required at various broadcasting bandwidths. Figure 6 shows the client buffer requirements regarding video length $L$, where the server bandwidth is varied from $3b$ to $12b$. The proposed scheme initially requires the largest buffering space. However, as the server bandwidth increases, the client buffer requirements drop and approach 50% of video size. Therefore, the proposed scheme yields smaller buffer requirements than AB and SingBroad.

5. Conclusion

A VoD system typically allocates a dedicated stream for each incoming video request; however, individual stream allocation easily causes the system overloaded. This study thus presents a fundamentally different approach by focusing solely on a class of applications identified as latency tolerant applications. Because video broadcasting does not provide interactive functions, a client is able to tolerate playback latency. One efficient broadcasting method is periodic broadcasting, which divides a video into smaller segments and broadcasts them periodically on multiple channels. However, the implementation of multiple-channel broadcasting is difficult and complicated. Therefore, this study proposes a novel single-channel broadcasting scheme for more efficient video delivery. The correctness of the scheme is verified mathematically. The performance comparisons show that, with the same settings of broadcasting bandwidth, the proposed scheme yields the shortest waiting time when compared with AB, HI, SingBroad, PAS, and ROS.

Acknowledgment

This work was financially supported by National Science Council, Taiwan, under a Research Grant no. NSC 101-2221-E-152-004.

References

frequency-splitting scheme for broadcasting hot videos in VOD
service," IEEE Transactions on Communications, vol. 50, no. 8,

casting (RFB) for video-on-demand applications,” IEEE Trans-

and waiting time for video-on-demand applications,” IEEE Trans-

with small latency and buffer demand for near video on
demand,” International Journal of Digital Multimedia Broadcast-

demand,” in Proceedings of the 10th International Conference on
Computer Communications and Networks (ICCCN ’01), pp. 418–

[13] H. F. Yu, P. H. Ho, and H. C. Yang, “Generalized sequence-
based and reverse sequence-based models for broadcasting hot
165, 2009.

casting scheme with live video support,” International Journal of

scheme for continuous media data broadcasting with a single

scheme considering units to play continuous media data," IEEE

for broadcasting continuous multimedia data over a single

broadcasting a video over a single channel," IET Communications,

scheduling scheme for broadcasting continuous multimedia
data over a single channel," IEEE Transactions on Broadcasting,

Submit your manuscripts at
http://www.hindawi.com