A New Fixed-Delay Broadcasting Protocol for Near Video-on-Demand Services

YU-WEI CHEN
Department of Computer and Information Science
Aletheia University
No. 32, Chen-Li Street, Tamsui, Taipei, 25103 Taiwan
REPUBLIC OF CHINA
http://www.cis.au.edu.tw

Abstract: - This work presents a novel fixed-delay broadcasting protocol for near video-on-demand service. Consider a film $S$ is divided into $n$ equal parts $\{S_1, S_2, ..., S_n\}$ and played through $k$ channels. Each segment $S_i$ must appear at least once every $i+c-1$ segments rather than $i$ segments. The value of parameter $c$ means that the users have to wait the time of $c$ segments. Each channel of $k$ channels is first partitioned into subchannels by a heuristic strategy. Then, a greedy approach is applied to assign the $n$ segments to $k$ channels. The waiting time is $c$ times the duration of one segment. As a result, the proposed method outperforms the previous broadcasting methods in terms of the maximum waiting time, for the same number of channels.

Key-Words: - Broadcasting, Fixed-delay, Multimedia, Near Video-on-Demand, Protocol, Waiting Time.

1 Introduction

The so-called Video-on-Demand (VOD) system refers to the use of a combination of a television set and a set-top-box or computer, lined via the Internet to a film provider. The system allows the user to choose which film to watch and utilize video-cassette-recorder capabilities such as playback, forward, rewind, pause and others. One of the simplest ways to utilize the various video-cassette-recorder capabilities is for the provider to set up a dedicated channel for the user when a request to view a film is made. This VOD system is called the true VOD system.

The main problem of the True VOD system lies in the requirement for an extremely large bandwidth for transmitting a large amount of information. Many researchers have proposed various solutions to reduce the need for bandwidth. They can be broadly categorized into three types for VOD systems – batching [1]-[3], stream tapping [4]-[6] or patching [7] and broadcasting [8]-[21].

Of the three VOD types, broadcasting saves much more bandwidth than the other two. The broadcasting approach works by first dividing the film into several segments, and then allowing the server to play them repeatedly in a few specified channels. Should a user wish to view the film, the greatest waiting time will be the duration of one such segment. Different methods [12]-[18] are used to divide the film into different numbers of segments, so waiting times differs. The user must also have enough buffer space to store the downloaded segments for continuous playback. This system is also known as the near VOD system.

Recently, Pâris et al. [22] and Chen [23] proposed “fixed-delay” broadcasting protocols. Should a user wish to view a film, the greatest waiting time will be the duration of $m$ segments rather than one segment. The two methods [22], [23] have shorter maximum waiting time than those in [12][15]-[18]. In addition, Pâris et al. [24] proposed another fixed-delay broadcasting protocol by using previews to reduce the cost of VOD services.

This work presents a novel fixed-delay broadcasting protocol for near VOD service. Consider a film $S$ is divided into $n$ equal parts $\{S_1, S_2, ..., S_n\}$ and played through $k$ channels. Each segment $S_i$ must appear at least once every $i+c-1$ segments rather than $i$ segments. Each channel of $k$ channels is first partitioned into subchannels by a heuristic strategy. Then, a greedy approach is applied to assign the $n$ segments to $k$ channels. The user simultaneously downloads the film segments from $k$ channels into the set-top-box (STB) and plays them in order. All users wait for a fixed delay $T$ before watching the video. This waiting time $T$ is $c$ times the duration of one segment. As a result, the proposed method outperforms the previous broadcasting methods [22], [23] in terms of the maximum waiting time, for the same number of channels.

This paper is organized as follows. In Section 2, preliminaries are broadly introduced. The new broadcasting protocol is proposed and analyzed in
Section 3. Finally, in Section 4, the conclusions are addressed.

2 Preliminaries

2.1 Fixed-Length Segment-Scheduling Problem

Fast Broadcasting [12][15], Pagoda broadcasting [16][17], and Recursive frequency splitting broadcasting [18] are all problems of Fixed-length Segment-Scheduling. In such a problem, the film $S$ is always divided into $n$ equal parts ($S_1, S_2, ..., S_n$) and played back through $k$ channels, at the same time assuming each channel's bandwidth is just sufficient to fit the usage rate of the film during normal playback and that the user can receive information from $k$ channels. Since before watching film segment $S_i$, the first $i-1$ segments $S_1, S_2, ..., S_{i-2}$, and $S_i$ have to be watched, segment $S_i$ has to at least appear once in the time every $i$ segments is played to ensure proper playback without breaks in between. Therefore segment $S_i$ at least needs to use up $1/i$ of channel bandwidth. Under the condition of uninterrupted playback, the largest number of segments that can be divided is $n$, and $n$ satisfies:

$$1 + \frac{1}{2} + ... + \frac{1}{n} = k < 1 + \frac{1}{2} + ... + \frac{1}{n+1}.$$

2.2 Traditional Broadcasting Protocols

2.2.1 Stagger Broadcasting

Assume we want to play a film $S$ of time length $L$ through $k$ channels $\{C_0, C_1, ..., C_{k-1}\}$. In the initial time, stagger broadcasting protocol [9] let the whole film will repeatedly transmit on $C_0$. After the time $iL/k$, $1 \leq i \leq k-1$, the same whole film will also be transmitted on $C_i$ periodically. The maximum waiting time is $L/k$.

2.2.2 Fast Broadcasting

The basic concept of fast broadcasting method [12] is described as follows. A film $S$ is first partitioned into $2^k-1$ segments $\{S_1, S_2, ..., S_{2^k-2}, S_{2^k-1}\}$. Segment $S_1$ will repeatedly transmit on $C_0$. The $2^i$ segments $\{S_{2^i}, S_{2^i+1}, ..., S_{2^{i+1}-1}\}$ repeatedly transmit in order on $C_i$ for $1 \leq i \leq k-1$.

2.2.3 Pagoda Broadcasting

The basic concept of pagoda broadcasting method [17] is described as follows. A film $S$ is first partitioned into $n$ segments $\{S_1, S_2, ..., S_n\}$ where $n = 4(5^{k-2}+1)-1$ if $k$ is even and $n = 2(5^{k/2})$ if $k$ is odd. Segment $S_i$ will repeatedly transmit on $C_i$. Let $q=2^r-1$ for $r \geq 1$ and the index of $S_i$ is smaller than those of the other segments broadcasted on $C_r$. The $z/2$ segments $\{S_{z/2}, S_{z/2+1}, ..., S_{z/2+1}\}$ repeatedly transmit in order on the even slots on $C_{z/2}$. The $z$ segments $\{S_{z}, S_{z+1}, ..., S_{z+1}\}$ repeatedly transmit in order on the odd even slots on $C_{z/2}$.

After transmitting segments on $C_{z/2}$, the following introduces the segments scheduling on $C_{z/2}$. The $z/2$ segments $\{S_{z}, S_{z+1}, ..., S_{z+1}\}$ repeatedly transmit in order on the slots $3i+1, 0 \leq i \leq z/2$. The $z$ segments $\{S_{z}, S_{z+1}, ..., S_{z+1}\}$ repeatedly transmit in order on the slots $3i+1, 0 \leq i \leq z/2$. The $z$ segments $\{S_{z}, S_{z+1}, ..., S_{z+1}\}$ repeatedly transmit in order on the slots $3i+1, 0 \leq i \leq z/2$.

Fig. 1 and Fig. 2 show the Pagoda scheme’s scheduling for three and four channels, respectively.

![Fig. 1. Pagoda scheme’s scheduling for three channels.](image)

![Fig. 2. Pagoda scheme’s scheduling for four channels.](image)

2.2.4 Recursive Frequency Splitting Broadcasting

The basic concept of recursive frequency splitting (RFS) broadcasting scheme [18] is described as follows. Tseng, Yang, and Chang first define the slot sequence $SS(C_r, p, q)$ as an infinite sequence of time slots $[p, p+q, p+2q, ...]$ belonging to channel $C_r$ beginning at slot $p$, and repeating infinitely with a period of $q$ slots, where $C_r$ is one of the $k$ channels, $p \geq 0$ is an integer, and $q \geq 1$ is an integer, $0 \leq p \leq q-1$.

Initially, let $POOL = \{SS(C_0, 0, 1), SS(C_1, 0, 1), SS(C_2, 0, 1), ..., SS(C_{k-1}, 0, 1)\}$ denote the set of free channels and let $j$ be the index of segment. The initial value of $j$ is 1. Second, pick a slot sequence $SS(C_r, p,$
\( q \) with the smallest value of \( j \) mod \( q \) from \( \text{POOL} \) such that \( q \leq j \). Let \( \text{POOL} = \text{POOL} - \{\text{SS}(C_i, p, q)\} \). Third, split \( \text{SS}(C_i, p, q) \) into \( \{\text{SS}(C_i, p, qa) \} \), \( \{\text{SS}(C_i, p+q, qa) \} \), \( \{\text{SS}(C_i, p+(a+1)q, qa) \} \), \ldots \( \{\text{SS}(C_i, p+(a-1)q, qa) \} \) where \( a = \lfloor j/q \rfloor \). Segment \( S_j \) is broadcast on the slots in \( \text{SS}(C_i, p, qa) \). Do the union \( \text{POOL} = \text{POOL} \cup \{\text{SS}(C_i, p+xq, qa) \} | 1 \leq x \leq a-1 \}. \) If \( \text{POOL} \) is not empty, then increase \( j \) by one and go to the second phase. Otherwise, terminate this process and output the value of \( j \). Fig. 3 illustrates the result of RFS algorithm with four channels.

\[
\begin{array}{c|cccccc}
\text{Channel} & c_1 & c_2 & c_3 & c_4 \\
\hline
\text{Subchannels} & S_1 & S_2 & S_3 & S_4 \\
\end{array}
\]

Fig. 3. Recursive frequency-splitting scheme’s scheduling for four channels.

### 2.3 Fixed-Delay Broadcasting Protocols

#### 2.3.1 Fixed-Delay Pagoda Broadcasting

Páris et al. [22] proposed a fixed-delay pagoda broadcasting (FDPB) protocol. With the FDPB protocol, segments \( S_i \) must be transmitted at least once per \( m+i-1 \) slots, where \( m \) is an integer \( m \geq 1 \). Consider a case in which \( m=9 \); the basic concept of FDPB protocol is thus described as follows. Segment \( S_1 \) is the first segment to be broadcast on channel \( C_0 \) and will need to be transmitted at least once every nine slots. The first channel \( C_0 \) is partitioned into three \((=\sqrt{9})\) subchannels. The three segments \( S_1, S_2 \) and \( S_3 \) will be assigned to the first subchannel on \( C_0 \) and each is repeated once per nine slots. The four segments \( S_4 \) to \( S_7 \) are assigned to the second subchannel on \( C_0 \) and each is repeated once per 12 slots. The five segments \( S_8 \) to \( S_{12} \) are assigned to the third subchannel on \( C_0 \) and each is repeated once per 15 slots.

Segment \( S_{13} \) is the first segment to be broadcast on channel \( C_1 \) and must be transmitted at least once per 21 \((=13+9-1)\) slots. The number 21 is not a square and the closest square is 25=\(5^2\), so channel \( C_1 \) is partitioned into five subchannels. Segments \( S_{13} \) to \( S_{16} \) are assigned to the first subchannel on \( C_1 \). Similarly, the four groups, Segments \( S_{17} \) to \( S_{21} \), Segments \( S_{22} \) to \( S_{27} \), Segments \( S_{28} \) to \( S_{34} \), and Segments \( S_{35} \) to \( S_{42} \), are assigned to the other four subchannels on \( C_1 \) while each of the segments of the four groups will be repeated once every 25, 30, 35 and 40 slots, respectively.

Segment \( S_{43} \) is the first segment to be broadcast on channel \( C_2 \) and must be transmitted at least once per 51 \((=43+9-1)\) slots. Therefore, channel \( C_2 \) is partitioned into seven subchannels and Segments \( S_{43} \) to \( S_{51} \) are assigned to \( C_2 \). The FDPB protocol can achieve segment-to-channel mapping for any \( k \) channels when the above concept is applied repeatedly. Table 1 illustrates the result of FDPB algorithm with seven channels for \( m=9 \).

<table>
<thead>
<tr>
<th>Channel</th>
<th>Number of Subchannels</th>
<th>First Segment</th>
<th>Last Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_0 )</td>
<td>5</td>
<td>( S_1 )</td>
<td>( S_{12} )</td>
</tr>
<tr>
<td>( C_1 )</td>
<td>5</td>
<td>( S_{13} )</td>
<td>( S_{42} )</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>7</td>
<td>( S_{43} )</td>
<td>( S_{116} )</td>
</tr>
<tr>
<td>( C_3 )</td>
<td>11</td>
<td>( S_{117} )</td>
<td>( S_{508} )</td>
</tr>
<tr>
<td>( C_4 )</td>
<td>18</td>
<td>( S_{109} )</td>
<td>( S_{814} )</td>
</tr>
<tr>
<td>( C_5 )</td>
<td>29</td>
<td>( S_{115} )</td>
<td>( S_{168} )</td>
</tr>
<tr>
<td>( C_6 )</td>
<td>47</td>
<td>( S_{2169} )</td>
<td>( S_{8180} )</td>
</tr>
</tbody>
</table>

### 2.3.2 Enhanced Recursive Frequency Splitting Broadcasting

The basic concept of enhanced recursive frequency splitting (ERFS) broadcasting scheme [23], called \( c \)-ERFS, is described as follows. The number of available channels is \( k \). Segment \( S_i \) must appear once per \( c \) segments, rather than per one segment. The \( c \)-ERFS uses the definition of the slot sequence \( \text{SS}(C_i, p, q) \) in [18].

Initially, let \( \text{POOL} = \{\text{SS}(C_i, p, 1) \} | 0 \leq i \leq k-1 \} \) denote the set of free channels. Each channel is partitioned into \( \lceil \sqrt{c} \rceil \) subchannels. Thus, \( \text{POOL} = \{\text{SS}(C_i, \delta, \lceil \sqrt{c} \rceil) \} | 0 \leq i \leq k-1, 0 \leq \delta < \lceil \sqrt{c} \rceil \rceil \}. \) The initial value of \( j \) is \( c \) for \( c \geq 2 \). Second, pick a slot sequence \( \text{SS}(C_i, p, q) \) with the smallest value of \( j \) mod \( q \) from \( \text{POOL} \) such that \( q \leq j \). Let \( \text{POOL} = \text{POOL} - \{\text{SS}(C_i, p, q) \} \). Third, split \( \text{SS}(C_i, p, q) \) into \( \{\text{SS}(C_i, p+xq, qa) \} | 0 \leq x \leq a-1, a = \lfloor j/q \rfloor \rceil \}. \) Segment \( S_{i+1} \) is broadcast on the slots in \( \text{SS}(C_i, p, qa) \). Set \( \text{POOL} = \text{POOL} \cup \{\text{SS}(C_i, p+xq, qa) \} | 1 \leq x \leq a-1 \}. \) If \( \text{POOL} \) is not empty, then increase \( j \) by one and go to second phase. Otherwise, terminate this process and output the value of \( j-c+1 \). Fig. 4 shows the example schedules of segments obtained using 2-ERFS with three channels.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Time Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( S_1 ) S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12}</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>( S_1 ) S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12}</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>( S_1 ) S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12}</td>
</tr>
</tbody>
</table>

Fig. 4. Schedule of segments for 2-ERFS with three channels.
3 Proposed Broadcasting Protocol

The basic concept of the proposed fixed-delay recursive-frequency-splitting broadcasting (FDRFS) protocol is described as follows. Each $S_i$ does not need to appear at least once per $j$ segments; rather, each $S_i$ must appear at least once per $j+c-1$ segments for $j, c \geq 1$. Before scheduling the segments on available channels, each channel is first divided into different subchannels. Letting $x=c$, channel $C_0$ is divided into $\lceil \sqrt{x} \rceil$ subchannels. Letting $x=x+\lceil \sqrt{x} \rceil$, channel $C_1$ is divided into $\lceil \sqrt{x} \rceil$ subchannels. Recursively, each channel is divided into different subchannels.

Then, the same concept as in [23] is improved and applied to schedule the segments, increasing the number of cut video segments. The waiting time for the user decreases accordingly. The difference between this paper and [23] is the channel-splitting strategy. For example, a slot sequence $SS(C, p, q)$ with the smallest value of $j$ mod $q$ is selected for $S_{j+c-1}$, $q \leq j$. The prime factorization of $j/q$ is $a_1*a_2*...a_m$ for $a_i \leq a_i$, $1 \leq i \leq m$. The slot sequence $SS(C, p, q)$ is first split into $SS(C, p+xq, a_iq) \{0 \leq x \leq a_i-1\}$. Then, $SS(C, p+(a_1-1)q, a_iq)$ is split into $SS(C, p+x(a_1-1)q, a_iq) \{0 \leq x \leq a_2-1\}$. Further, $SS(C, p+(a_1-1)(a_2-1)q, a_iq)$ is split into $SS(C, p+x(a_1-1)(a_2-1)q, a_iq) \{0 \leq x \leq a_1-1\}$. Recursively do the splitting processing, finally, $SS(C, p+(a_1-1)(a_2-1)...(a_{m-1}-1)q, a_iq) \{0 \leq x \leq a_{m-1}-1\}$. Consequently, $SS(C, p, q)$ can be split into $SS(C, p+xq, a_iq) \{0 \leq x \leq a_i-1\}$. Letting $x=x+\lceil \sqrt{x} \rceil$ and $i=i+1$. If $i < k-1$, then redo Step 2. Otherwise, go to Step 3.

Step 3. Set $j=c$ for $c \geq 1$. Segment $S_{c-1}$ must appear at least once per $j$ segments.

Step 4. Select a slot sequence $SS(C, p, q)$ with the smallest value of $j$ mod $q$ from $POOL$ such that $q \leq j$. Let $POOL = POOL - \{SS(C, p, q)\}$.

Step 5. Let the prime factorization of value $j/q$ as $a_1*a_2*...a_m$. Split $SS(C, p, q)$ into $SS(C, p+xq, a_iq) \{0 \leq x \leq a_i-1\}$.

Step 6. If $POOL$ is not empty, then increase $j$ by one and go to Step 4. Otherwise, terminate this process and output the value of $j+c-1$.

Since segment $S_j$ must appear at least once per $j+c-1$ slots to ensure proper playback without breaks, segment $S_j$ at least needs to use up $1/(i+c-1)$ of channel bandwidth. For uninterrupted playback, the largest number of segments that can be divided is $n$, and the upper bound on $n$ satisfies:

$$\frac{1}{c} + \frac{1}{c+1} + ... + \frac{1}{n} = k < \frac{1}{c} + \frac{1}{c+1} + ... + \frac{1}{n+1}.$$

Table 2 compares previous FDPB [22], ERFS [23], the proposed FDRFS protocol and the upper bound in terms of the total segments. In the proposed protocol, each $S_i$ must appear at least once per $j+c-1$ segments for $j, c \geq 1$. Therefore, the user must wait for the time between $c-1$ and $c$ segments to ensure proper playback without breaks. For a fair comparison, the total number of segments is divided by $c$, whose division can be thought of as indicating that $c$ segments are packed into a large segment. Table 3 compares previous methods and the proposed method in terms of the total segments. The proposed method clearly outperforms the previous methods in terms of the longest waiting time.
Like the following methods [12][16]-[18][22][23], the client requires buffer to store a portion of the video. The bound of the buffer requirement can be analysed by applying the same analysis concept [25]. Consider a film $S$ is divided into $n$ equal segments and played through $k$ channels. Each segment $S_i$ must appear at least once every $i+c-1$ time slots. Since client receives $k$ segments every time slot but consuming only one segment and segment $S_j$ is not to buffer after watching, the maximum buffer requirement must be bounded by:

$$B_k = \max_{i \geq c} \left\{ \left[ k \cdot (i+c-1) - \sum_{j=1}^{i+c-1} \frac{1}{p_{k,j}} \right] \right\}$$

where $p_{k,j}$ denotes the time period of the slot sequence assigned to the segment $S_j$ when using $k$ channels.

4 Conclusion

This work has presented a fixed-delay recursive-frequency-splitting broadcasting protocol for near video-on-demand service. The proposed method outperforms the previous broadcasting methods in terms of the maximum waiting time, for the same number of channels. The bound of the maximum buffer requirement is analysed too.

References:


