Research Article

Efficient Periodic Broadcasting for Mobile Networks at Small Client Receiving Bandwidth and Buffering Space

Hsiang-Fu Yu, 1 Yao-Tien Wang, 2 Jong-Yih Kuo, 3 and Chu-Yi Chien 1

1 Department of Computer Science, National Taipei University of Education, Taipei 10671, Taiwan
2 Department of Computer Science and Information Engineering, Hungkuang University, Taichung 43302, Taiwan
3 Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei 10608, Taiwan

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

Received 7 January 2013; Accepted 18 March 2013

Copyright © 2013 Hsiang-Fu Yu et al. 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.

Periodic broadcasting is an effective approach for delivering popular videos. In general, this approach does not provide interactive (i.e., VCR) functions, and thus a client can tolerate playback latency from a video server. The concept behind the approach is partitioning a video into multiple segments, which are then broadcast across individual communication channels in terms of IP multicast. The method improves system throughput by allowing numerous clients to share the channels. For many broadcasting schemes, client receiving bandwidth must equal server broadcasting bandwidth. This limitation causes these schemes to be infeasible in mobile networks because increasing receiving bandwidth at all client sites is expensive, as well as difficult. To alleviate this problem, the fibonacci broadcasting (FiB) scheme allows a client with only two-channel bandwidth to receive video segments. In comparison with other similar schemes, FiB yields smallest waiting time. Extending FiB, this work proposes a new scheme (called FiB+) to achieve smaller client buffering space and the same waiting time under two-channel receiving bandwidth. Extensive analysis shows that FiB+ can yield 34.5% smaller client buffer size than that of FiB. Further simulation results also indicate that FiB+ requires lower client buffering space than several previous schemes.

1. Introduction

Video-on-demand (VOD) services have become popular due to advances in network and computer technology [1, 2]. A VOD system may easily run out of bandwidth since the growth in bandwidth can never keep up with the growth in the number of clients. To alleviate this problem, one way is to simply broadcast popular videos. According to the study in [3, 4], a few very popular videos constitute most client requests. Data broadcasting is thus suitable to transfer popular videos that may be interesting to many clients in a particular period of time. An efficient method for broadcasting a popular video is to divide it into segments, which are simultaneously and periodically transmitted across individual communication channels in terms of IP multicast [5]. Because video broadcasting does not provide VCR functions, a client is able to tolerate playback latency. To ensure continuous playback, clients must simultaneously download and save the video segments from these channels. The clients usually have to wait for the occurrence of the first segment before they can start playing the video. Since the clients cannot watch the video immediately, the broadcasting schemes provide near VOD services.

The fast broadcasting (FB) scheme [6] improves segment partition and arrangement to yield shorter waiting time. To achieve near-minimum waiting time, the recursive frequency-splitting (RFS) scheme [7] broadcasts each segment at the frequency that can keep continuous video playback. A scalable binomial broadcasting scheme [8] transfers a variable-length video using constant bandwidth. To simplify the implementation of multiple channels, the PAS scheme [9] broadcasts video segments over a single channel. The reverse-order scheduling (ROS) scheme [10] transmits segments of the same group in reverse order over a single channel to save buffering space.
With the fast growth of wireless networks, mobile video services become more and more popular. Broadcasting videos under rather restricted client resources is increasingly important. The following schemes address the savings on client buffer size and bandwidth. Modifying the FB scheme [6], the reverse fast broadcasting (RFB) scheme [11] buffers 25% of video size, just half of what is required by FB. By combining RFs and RFB, the hybrid broadcasting scheme (HyB) [12] achieves small client buffering space and waiting time. Different RF-based hybrid schemes were proposed in [13, 14]. The skyscraper broadcasting (SkB) scheme [15] allows a client to download video data using only a bandwidth of two channels. The client-centric approach (CCA) [16] also permits a client downloading video data via a small number of channels, and CCA+ [17] further yields smaller waiting time than SkB. Like SkB and CCA+, the fibonacci broadcasting (FiB) scheme [18] supports a client with two-channel bandwidth but achieves the minimum waiting time. The authors in [19] proposed an FB-based scheme for heterogeneous clients. The studies in [20, 21] deploy a proxy in VoD systems to serve heterogeneous clients.

The contributions of this study are summarized as follows.

1. Extending FiB, this work proposes a promising scheme, called FiB+, to deliver near VOD services to clients with small receiving bandwidth and buffering space. In comparison with FiB, FiB+ still yields the minimum waiting time under two-channel receiving bandwidth; moreover, FiB+ can save about 34.5% of buffer size.

2. The paper investigates the total client buffer requirements for FiB+ and explains why this scheme requires smaller buffering space than FiB. We further derive the maximum number of segments buffered by an FiB+ client mathematically. Extensive performance analysis has been conducted on FiB+ by comparing a number of past reported counterparts. The results indicate that FiB+ yields relatively lower client buffer requirements than most schemes.

The remainder of this study is organized as follows. The FiB scheme is introduced in Section 2. Section 3 presents FiB+. This section also verifies the on-time video delivery under two-channel client bandwidth. Section 4 evaluates the performance of FiB+. Brief conclusions are drawn in Section 5.

2. Review of FiB

Let $k$ be the number of server channels throughout the paper. The FiB scheme [18] unequally divides a video of length $L$ into $k$ segments, denoted by $S_1, S_2, \ldots, S_k$ in sequence. The length of a segment $S_i$ is based on the following equation and equals $L n_i / \sum_{p=1}^{k} n_p$ as follows:

$$n_i = \begin{cases} 1, & i = 1 \\ 2, & i = 2 \\ n_{i-1} + n_{i-2}, & 3 \leq i \leq k \end{cases}$$

Assume that the data transmission rate of each channel equals the playback rate $b$. The server then periodically broadcasts segment $S_i$ on channel $C_i$, as illustrated in Figure 1. In the figure, segments downloaded and played by a client are gray. When a client wants to watch a video, the client first downloads segments $S_1$ and $S_2$ on the first two channels $C_1$ and $C_2$. Once finishing receiving the segment $S_1$, the client continuously accepts segment $S_2$ and newly downloads segment $S_3$ on channel $C_3$. The client repeats the process, which starts downloading segment $S_1$ on channel $C_1$ once finishing receiving segment $S_{i-2}$ from channel $C_{i-2}$, until all the segments are loaded. An FiB client thus requires a bandwidth of only two channels to download video segments.

### Table 1: List of terms used in the proposed scheme 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>$k$</td>
<td>Number of broadcasting channels for each video on the server side</td>
</tr>
<tr>
<td>$C_i$</td>
<td>$i$th broadcasting channel, $i = 1, \ldots, k$</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of segments of a video</td>
</tr>
<tr>
<td>$S_i$</td>
<td>$i$th video segment, $i = 1, \ldots, N$</td>
</tr>
<tr>
<td>$n_i$</td>
<td>Number of segments transferred on channel $C_i$</td>
</tr>
<tr>
<td>$m_i$</td>
<td>Total segments transferred on channels $C_1$ through $C_i$, $m_i = \sum_{p=1}^{i} n_p$</td>
</tr>
<tr>
<td>$b$</td>
<td>Video playback rate assumed to equal the data transmission rate of each channel</td>
</tr>
<tr>
<td>$T_0$</td>
<td>Starting time to watch the first segment</td>
</tr>
<tr>
<td>Time unit</td>
<td>Basic unit on time axis, whose length equals the length of a segment (i.e., $L/N$)</td>
</tr>
</tbody>
</table>

Some of the frequently used terms and their definitions are listed in Table 1. On the server side, the FiB+ scheme includes the following steps.

1. The server equally divides a video into $N$ segments, denoted by $S_1, S_2, \ldots, S_N$ in sequence, where $N = \sum_{p=1}^{k} n_p$, where $n_p$ is based on (1). The length of each segment thus equals $L/N$. From (1), we further yields

$$m_i = \sum_{p=1}^{i} n_p = n_{i+2} - 2.$$  (2)

See Appendix A for details. Thus, $N = n_{k+2} - 2$. The FiB+ scheme then assembles segments $S_{n_{k+2}}$ to $S_{n_{k+2} - 2}$ (i.e., $S_{m+1}$ to $S_m$) into group $G_1$ sequentially, as illustrated in Figure 2(a). The number of segments of group $G_1$ thus equals $n_k$. For instance, group $G_4$ includes segments $S_7$ to $S_{11}$, and $n_k = 5$.

2. Channel $C_i$, where $1 \leq i \leq k - 2$, periodically broadcasts the segments of group $G_i$ in sequence, as shown in Figure 2(b). That is, segments $S_{n_{i+1}}$ to $S_{n_{i+2}}$ are transferred one by one on channel $C_i$. 

$$m_i = \sum_{p=1}^{i} n_p = n_{i+2} - 2.$$  (2)
(3) The segments of groups \(G_{k-1}\) and \(G_k\) are cyclically transmitted on channels \(C_{k-1}\) and \(C_k\) in reverse order, respectively. Figure 2(b) shows that the scheme repeatedly broadcasts the segments of group \(G_{k-1}\) in the order of \(S_{n_{k-1}}\) to \(S_{n_k-1}\) and the segments of group \(G_k\) in the order of \(S_{n_k}\) to \(S_{n_k+1}\).

Figure 3 demonstrates the segment broadcasting and downloading for FiB+, where the segments downloaded and played by a client are gray. Let \(T_0\) be the time that the client starts receiving video segments and be the origin (i.e., the first time unit) of the time axis. Due to \(k = 6\), FiB+ equally divides a video into 32 segments, which are then classified into six groups. The segments of groups \(G_1\) to \(G_4\) are broadcast sequentially on channels \(C_1\) to \(C_4\), respectively. In addition, FiB+ transmits segments of groups \(G_5\) and \(G_6\) on channels \(C_5\) and \(C_6\) in reverse order.

A client is assumed to have enough buffers to store video segments downloaded. We further suppose that one time unit equals the length of a segment throughout this study. Playing a video on the client side includes the following steps.

(1) Download segments of group \(G_i\) on channel \(C_i\) during time units \(n_{i-1}\) to \(n_{i-1} + n_i - 1 = n_{i+1} - 1\), where...
1 \leq i \leq k - 2 \) and \( n_0 = 1 \). For example, the client accepts segments from channel \( C_4 \) during time units \( n_{k-1} = 3 \) to \( n_{k+1} - 1 = 7 \), as shown in Figure 3.

(2) The paper next presents how a client receives segments from channels \( C_{i-1} \) and \( C_i \). (In Figure 3, refer to channels \( C_3 \) and \( C_6 \).) Suppose that a client first sees a segment \( S_x \) on a channel \( C_i \) at time \( T_{\text{now}} \), and sees the next segment \( S_y \) at time \( T_{\text{next}} \), where \( i = k - 1 \) or \( i = k \). (In Figure 3, \( i = 6 \) and \( y = 25 \).) The client is also assumed to play segments \( S_x \) and \( S_y \) at time \( T_{\text{now}} \) and \( T_{\text{next}} \), respectively. Clearly, if \( T_{\text{next}} \leq T_{\text{use}} \), the client can delay downloading segment \( S_y \) at time \( T_{\text{now}} \), without interrupting the playback. Substituting \( T_{\text{use}} = y \) th time unit, \( T_{\text{next}} = x \) th time unit, and \( T_{\text{next}} = T_{\text{now}} + n_i \) into \( T_{\text{next}} \leq T_{\text{use}} \), we obtain

\[ x + n_i \leq y. \tag{3} \]

If the inequality is true, the client does not receive segment \( S_y \) at time \( T_{\text{now}} \), otherwise, performs the downloading immediately. For instance, when the client first sees segment \( S_4 \) on channel \( C_6 \) at the 7th time unit (i.e., \( T_{\text{now}} \)) in Figure 3, (3) is true, \( 7 + 13 \leq 25 \), and the client does not download the segment. Afterwards, the client sees next segment \( S_5 \) with gray color and diagonal lines at the 20th time unit. The client must receive the segment because (3) does not hold, \( 20 + 13 > 25 \).

(3) The client plays the video in the order of \( S_1, S_2, \ldots, S_N \) at time \( T_{\text{now}} \).

(4) The client stops loading data from networks when all the segments have been received.

FiB+ and FiB differ in three areas.

(i) Equal-length segment partition versus variable-length segment partition. FiB+ divides a video into multiple equal-length segments, while FiB partitions a video into variable-length segments. For example, given \( k = 6 \), FiB divides a video into six segments, whose lengths are \( \frac{L}{32}, \frac{2L}{32}, \frac{3L}{32}, \frac{4L}{32}, \frac{5L}{32}, \frac{6L}{32}, \) and \( \frac{7L}{32} \). On the other hand, FiB+ partitions a video into 32 segments, whose lengths all equal \( \frac{L}{32} \).

(ii) Multiple segments on each channel versus single segment. FiB+ cyclically broadcasts several segments on each channel except the first channel, and FiB transmits only one.

(iii) Segment transmission in reverse order. The FiB+ scheme broadcasts segments on the last two channels in reverse order. For example, the scheme transmits segments \( S_{32} \) to \( S_{4} \) on channel \( C_2 \) and segments \( S_{31} \) to \( S_{20} \) on channel \( C_6 \), as illustrated in Figure 3.

3.1. Analysis of Segment Playing and Downloading on a Single Channel. We next analyze the segment downloading on channel \( C_i \), where \( 1 \leq i \leq k \).

For \( 1 \leq i \leq k - 2 \), a client receives segments \( S_{n_{i-1}+1} \) to \( S_{n_{i+2}-2} \) from channel \( C_i \) during time units \( n_{i-1} \) to \( n_{i+1} - 1 \) and plays the segments during time units \( n_{i+1} - 1 \) to \( n_{i+2} - 2 \), as mentioned previously. Suppose that a client sees a segment \( S_j \) at the \( (n_{j+1} - 1) \) th time unit, where \( n_{j+1} - 1 \leq j \leq n_{i+2} - 2 \). Figure 4(a) shows how a client downloads and plays segments, where the segments downloaded and played by the client are gray.

For \( i = k - 1 \) or \( k \), a client downloads segments according to (3). We also assume that a client sees a segment \( S_j \) at the \( (n_{j+1} - 1) \) th time unit, where \( n_{j+1} - 1 \leq j \leq n_{i+2} - 2 \). A complete segment-downloading diagram for channel \( C_i \) is based on (3), as indicated in Figure 4(b). The explanation is as follows.

The client always downloads segment \( S_j \) since the inequality of (3) does not hold for \( x = n_{i+1} - 1 \) and \( y = j \) (i.e., \( n_{i+1} - 1 + n_i = n_{i+2} - 1 > j \)). In addition, because the segments of group \( G_i \) are transmitted once on channel \( C_i \) every \( n_i \) time units and are played during time units \( n_{i+1} - 1 \)
to $n_{i+2} - 2$, the client downloads a segment of group $G_i$ either during time units $n_{i+1} - 1$ to $n_{i+2} - 2$ or during time units $n_{i-1}$ to $n_{i+1} - 2$ (i.e., before the downloading of segment $S_j$).

3.1. Segment Downloading within $[n_{i-1}, n_{i+2} - 2]$. Figure 4(b) shows that segments $S_{n_{i-1}}$ to $S_i$ are broadcast on channel $C_i$ in descending order during time units $n_{i+1} - 1$ to $j$, while a client plays these segments in turn. Let $S_a$ be a segment broadcast on channel $C_i$ during this period, and let $S_b$ be the client-playback segment corresponding to $S_a$. Clearly, if $a \geq b$, a client must download segment $S_a$ to ensure continuous playing. Because the number of segments between $S_j$ and $S_a$ on channel $C_i$ equals the number of segments between $S_{n_{i-1}}$ and $S_b$ on the client playback, $j - a = b - (n_{i+1} - 1)$ and $a = j + n_{i+1} - 1 - b$. Substituting this equation to $a \geq b$ yields $(j + n_{i+1} - 1)/2 \geq b$. The maximum value of $b$ is $(j + n_{i+1} - 1)/2$, and the corresponding value of $a$ equals $j + n_{i+1} - 1 - [(j + n_{i+1} - 1)/2] = (j + n_{i+1} - 1)/2$. Figure 4(b) illustrates that the client downloads segments $S_j$ to $S_{(j+n_{i+1} - 1)/2}$ during time units $n_{i+1} - 1$ to $(j + n_{i+1} - 1)/2$ and does not download any segment during time units $(j + n_{i+1} - 1)/2 + 1$ to $j$. Similarly, this study obtains that a client receives segments $S_{n_{i+2} - 2}$ to $S_{(j+n_{i+1} - 1)/2}$ during time units $j + 1$ to $(j + n_{i+2} - 1)/2$ and does not download any segment during time units $(j + n_{i+2} - 1)/2 + 1$ to $n_{i+2} - 2$.

3.1.2. Segment Downloading within $[n_{i-1}, n_{i+2} - 2]$. From (3), the client must download segments $S_{(j+n_{i+1} - 1)/2} + 1$ to $j$ during time units $(j + n_{i+1} - 1)/2 + 1 - n_i$ to $n_{i+2} - 2$ because the client does not download these segments during time units $(j + n_{i+1} - 1)/2 + 1$ to $j$. For FiB+, the segment broadcasting frequency is large enough to let clients receive video data in time.

3.2. Workable Verification. This section shows that FiB+ guarantees continuous playback and two-channel bandwidth demand on the client side.

3.2.1. Continuous Playback. To keep on-time video delivery, the study in [7] indicates that a video server must broadcast a segment $S_j$ on a channel $C_i$ at least once in every $j$ time units. For FiB+, a server transmits a segment $S_j$ on every $n_i$ time units, where $n_{i+1} - 1 \leq j \leq n_{i+2} - 2$. This paper thus needs to prove $j \geq n_i$. We then evaluate

\[ j - n_i \geq (n_{i+1} - 1) - n_i \quad \text{due to } n_{i+1} - 1 \leq j \leq n_{i+2} - 2 \]

\[ = n_{i+1} - 1 - n_i \geq 0 \]

For FiB+, the segment broadcasting frequency is large enough to let clients receive video data in time.

3.2.2. Two-Channel Bandwidth Demand. From the previous analysis in Figure 4, we make a temporal-channel map of segment downloading for each channel, as indicated in Figure 5. In this figure, segments downloaded and played by a client are gray. This work divides client playback time $t$ (in terms of time units) into multiple successive durations for ease of explanation.

For $T_0 \leq t \leq n_{k-2} - 1$, the client merely receives segments from channels $C_1$ to $C_{k-2}$ because the client starts the segment
downloading on channel $C_{k-1}$ at time unit $n_{k-1}$, as shown in Figure 5. According to Step 1 of segment downloading on the client side, the client uses two-channel bandwidth to load segments in this period.

For $n_{k-2} \leq t \leq n_{k-1} - 1$, the client receives segments only from channels $C_{k-2}$ and $C_{k-1}$ because the client finishes receiving segments from channel $C_1$ to $C_{k-3}$ before time unit $n_{k-2}$ and starts downloading segments on channel $C_k$ at time unit $n_{k-1}$, as indicated in Figure 5.

For $n_{k-1} \leq t$, the client simply receives the remaining segments from channels $C_{k-1}$ and $C_k$ because the segment downloading on channel $C_{k-2}$ completes at time unit $n_{k-1} - 1$.

Accordingly, an FiB+ client can download segments using two-channel bandwidth.

4. Performance Analysis and Comparison

When a client just misses segment $S_1$ of a requested video, the maximum waiting time $\delta$ equals the access time of the segment from the first channel. Thus, $\delta = L/N = L/\sum_{p=1}^{k} n_p$, the same as that of FiB. According to the previous studies [15, 17], this work has calculated the values of $N$ offered by these schemes at different numbers of channels, as listed in Table 2. The larger the value is, the smaller the waiting time is. The table reveals that FiB and FiB+ yield far bigger values than other schemes. Figure 6 shows maximum waiting time versus server channels. FiB and FiB+ thus achieve much smaller waiting time than SkB and CCA+ under two-channel client bandwidth. For example, when the server bandwidth equals 10 channels, FiB and FiB+ reduce the broadcast latency to less than 32 seconds. By contrast, SkB and CCA+ incur more than 51 and 48 seconds, respectively.

Before analyzing the entire buffer requirements, we first investigate the number of the buffered segments when a client performs segment downloading on a single channel $C_i$.

Lemma 1. Let $B(i,t)$ be the function of the number of the segments buffered by an FiB+ client on channel $C_i$ at the $t$th time unit.

For $1 \leq i \leq k-2$,

$$B(i,t) = \begin{cases} 0, & t < n_{i-1}, \\ t - n_{i-1} + 1, & n_{i-1} \leq t \leq n_{i+1} - 2, \\ 0, & n_{i+2} - 2 < t. \\ \end{cases}$$

(5a)

For $i = k-1$ or $k$,

$$B(i,t) = \begin{cases} 0, & t < n_{i-1}, \\ \left\lfloor \frac{t - n_{i-1} + 2}{2} \right\rfloor, & n_{i-1} \leq t \leq n_{i+1} - 2, \\ \left\lfloor \frac{n_i}{2} \right\rfloor, & n_{i+1} - 2 < t \leq n_{i+2} - 2 - \left\lfloor \frac{n_i}{2} \right\rfloor, \\ 0, & n_{i+2} - 2 < t. \\ \end{cases}$$

(5b)

Proof. See Appendix B.

Equation (5b) shows that a client buffers at most $\lceil n_i/2 \rceil$ segments from channel $C_i$ for $i = k-1$ or $k$. On the other hand, an FiB client needs to buffer $n_i - 1$ segments [18]. Such a difference leads to the result that FiB+ requires much smaller buffering space.

Theorem 2. Let $B(t)$ be the maximum number of segments buffered by an FiB+ client. Then, $B(t) \leq \lceil n_{k-1}/4 \rceil + \lceil n_k/2 \rceil$.

Proof. See Appendix C.

Due to $\lim_{i \to \infty} (n_{i+1}/n_i) \approx ((1 + \sqrt{5})/2)$ [22] and $N = n_{k+2} - 2$, $\lim_{k \to \infty} ((\lceil n_{k-1}/4 \rceil + \lceil n_k/2 \rceil)/N) \approx 0.25$. This result
Table 2: The values of N offered by different schemes.

<table>
<thead>
<tr>
<th>k</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiB/FiB+</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>19</td>
<td>32</td>
<td>53</td>
<td>87</td>
<td>142</td>
<td>231</td>
</tr>
<tr>
<td>SkB</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>27</td>
<td>39</td>
<td>64</td>
<td>89</td>
<td>141</td>
</tr>
<tr>
<td>CCA+2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>27</td>
<td>39</td>
<td>64</td>
<td>89</td>
<td>149</td>
</tr>
</tbody>
</table>

indicates that an FiB+ client can buffer only 25% of video size, like RFB. We used the Perl language to implement a simulator, which could estimate the buffer requirements for FiB+. The results are listed in Table 3. The buffer size required by FiB+ is quite close to the bound when k ≥ 6. Because FiB has to buffer \( n_k - 1 \) segments, we can derive the buffer reduction rate of FiB+ versus FiB as follows: \( \lim_{k \to \infty} \frac{(\lceil n_{k-1}/4 \rceil + \lfloor n_k/2 \rfloor)/(n_k - 1)}{n_k - 1} \approx 0.345 \). FiB+ can reduce buffer requirements by 34.5%, when compared with FiB. Table 3 shows that with the growth of \( k \), the reduction rate is close to the bound. For example, for \( k = 10 \), an FiB client buffers 38.1% of video size. By contrast, an FiB+ client buffers only 25.1%. The reduction rate is 34.1%. According to the previous studies [15, 17], this work presents the buffer sizes required by different broadcasting schemes at different numbers of server channels, as indicated in Figure 7. For \( k > 3 \), the FiB+ scheme outperforms all the schemes.

5. Conclusions

With the advance of mobile computing technology, many clients access VOD services through their mobile devices. Delivering videos under rather restricted client resources is increasingly important. To fulfill this requirement, several schemes, such as SkB, FiB, and CCA+, are proposed to allow a client to watch a video using two-channel bandwidth. Extending FiB, this work devises FiB+, which exhibits the merits of small client waiting time and buffering space. The scheme still guarantees on-time video delivery under two-channel receiving bandwidth. According to the performance analysis, FiB+ yields the minimum waiting time and requires smaller client buffer size, when compared with most existing schemes.

Appendices

A. Proof of Equation (2)

For \( i \mod 2 = 1 \), let \( i = 2q + 1 \). Then,

\[
\sum_{p=1}^{i} n_p = n_1 + n_2 + n_3 + n_4 + \cdots + n_i
\]

\[
= n_3 + n_5 + \cdots + n_{2q+1} + n_{2q+1}, \quad \text{from (1)}
\]

\[
= n_2 + n_3 + n_5 + \cdots + n_{2q+1} + n_{2q+1} - n_2
\]

\[
= n_4 + n_5 + \cdots + n_{2q+1} + n_{2q+1} - n_2, \quad \text{from (1)}
\]

\[
= n_{2q+2} + n_{2q+1} - n_2, \quad \text{from (1)}
\]

\[
= n_{2q+3} - n_2, \quad \text{from (1)}
\]

\[
= n_{i+2} - 2.
\]  

For \( i \mod 2 = 0 \), let \( i = 2q \). Then,

\[
\sum_{p=1}^{i} n_p = n_1 + n_2 + n_3 + n_4 + \cdots + n_i
\]

\[
= n_3 + n_5 + \cdots + n_{2q+1}, \quad \text{from (1)}
\]

\[
= n_2 + n_3 + n_5 + \cdots + n_{2q+1} - n_2
\]

\[
= n_4 + n_5 + \cdots + n_{2q+1} - n_2, \quad \text{from (1)}
\]
= n_{2q+2} - n_2, \quad \text{from (1)}
= n_{i+2} - 2.

(A.2)

Accordingly, \( \sum_{p=1}^{i} n_p = n_{i+2} - 2 \).

B. Proof of Lemma 1

This study first proves (5a). For easy understanding, please refer to Figure 4(a).

(1) For \( t < n_{i-1} \), a client downloads no segment on channel \( C_i \) because these segments will appear again during time units \( n_{i-1} \) to \( n_{i+1} - 1 \). Thus, \( B(i, t) = 0 \).

(2) For \( n_{i-1} \leq t \leq n_{i+1} - 2 \), a client continuously accepts one segment every time unit but consumes no segment. Thus, the number of buffered segment equals \( t - n_{i-1} + 1 \). When \( t = n_{i+1} - 2 \), the client buffers the maximum segments and \( B(i, t) = n_{i+1} - 1 \).

(3) For \( n_{i+1} - 2 < t \leq n_{i+1} - 2 \), Figure 4(a) shows that a client stops loading data but plays the video in this period. The client consumes one segment every time unit, and thus the buffered segments decrease, \( B(i, t) = n_i - (t - (n_{i+1} - 2)) = n_{i+2} - 2 - t \).

(4) For \( n_{i+1} - 2 < t \), a client has finished playing all the segments on channel \( C_i \), and thus \( B(i, t) = 0 \).

The proof for (5b) is as follows. For easy understanding, please refer to Figure 4(b).

(1) For \( t < n_{i-1} \), a client does not download any segment on channel \( C_i \), and thus \( B(i, t) = 0 \).

(2) For \( n_{i-1} \leq t \leq n_{i+3} - 2 \), the paper divides the value range of \( t \) into four successive subranges for ease of proof.

(a) For \( n_{i-1} \leq t \leq [(j + n_{i+1} - 1)/2] - n_i \), Figure 4(b) shows that the client downloads no segment on channel \( C_i \), and thus \( B(i, t) = 0 \).

(b) For \( [(j + n_{i+1} - 1)/2] - n_i < t \leq j - n_i \),

\[
B(i, t) = t - \left( \left\lfloor \frac{j + n_{i+1} - 1}{2} \right\rfloor - n_i \right), \quad \text{see Figure 4(b)}
\leq t + n_i - \left\lfloor \frac{(t + n_i) + (n_{i+1} - 1)}{2} \right\rfloor, \quad \text{due to } t \leq j - n_i
\leq \left\lceil \frac{t - n_{i+1} + 2}{2} \right\rceil.
\]

(B.1)

(c) For \( j - n_i < t \leq [(j + n_{i+2} - 1)/2] - n_i \),

\[
B(i, t) = (j - n_i) - \left( \left\lfloor \frac{j + n_{i+1} - 1}{2} \right\rfloor - n_i \right), \quad \text{see Figure 4(b)}
\leq \left\lceil \frac{t - n_{i+1} + 2}{2} \right\rceil, \quad \text{due to } j - n_i < t.
\]

(B.2)

Thus, \( B(i, t) \leq \left\lfloor \frac{t}{2} \right\rfloor \).

(3) For \( n_{i+1} - 2 < t \leq n_{i+2} - 2 \), the client plays one segment every time unit while downloading at most one segment. The number of buffered segments is not larger than that at time unit \( t = n_{i+1} - 2 \), and thus \( B(i, t) \leq n_{i+2} - 2 \).

(4) For \( n_{i+2} - 2 < [n_{i+2}/2] < t \leq n_{i+2} - 2 \), the client has played \( j - (n_{i+2} - 2) \) segments, and the number of the remaining segments is

\[
n_i - (t - (n_{i+1} - 2)) = n_{i+2} - 2 - t.
\]

Thus, \( B(i, t) \leq n_{i+2} - 2 - t \).

(5) For \( n_{i+2} - 2 < t \), the client finishes playing all the segments on channel \( C_i \) so \( B(i, t) = 0 \).

The proof is complete.

C. Proof of Theorem 2

This work divides client playing time \( t \) (in terms of time units) into multiple successive durations for ease of proof.

(1) For \( t \leq n_{k-2} - 2 \), Figure 5 shows that the client simply receives segments from channels \( C_1 \) to \( C_{k-2} \). In addition, the client downloads two segments but plays only one every time unit. The number of buffered segments thus increases with time and achieves the maximum at time unit \( n_{k-2} - 2 \). At this time, the client has finished playing segments from channels \( C_1 \) to \( C_{k-4} \), and thus simply buffers segments from channels \( C_{k-3} \) and \( C_{k-2} \). Accordingly,

\[
B(t) = B(k - 3, n_{k-2} - 2) + B(k - 2, n_{k-2} - 2)
= \left( n_{k-2} - 2 \right) - n_{k-4} + 1
+ \left( n_{k-2} - 2 \right) - n_{k-3} + 1, \quad \text{from (5a)}
\]

(5) For \( n_{k-2} - 2 < t \leq n_{k-1} - 2 \), the client has finished playing all the segments received from channels \( C_1 \) to \( C_{k-4} \) and downloads no segment from channel \( C_k \). We thus merely consider the segments on channels \( C_{k-3} \) to \( C_{k-1} \). The client downloads at least one segment from channel \( C_{k-2} \) but plays
Table 3: Comparison of buffering space in the percentage of video size using $k$ server channels.

<table>
<thead>
<tr>
<th>$k$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiB (%)</td>
<td>0</td>
<td>33.3</td>
<td>33.3</td>
<td>36.4</td>
<td>36.8</td>
<td>37.5</td>
<td>37.7</td>
<td>38</td>
<td>38</td>
<td>38.1</td>
</tr>
<tr>
<td>FiB+ (%)</td>
<td>0</td>
<td>33.3</td>
<td>33.3</td>
<td>27.3</td>
<td>26.3</td>
<td>25</td>
<td>24.5</td>
<td>25.3</td>
<td>25.4</td>
<td>25.1</td>
</tr>
<tr>
<td>Reduction rate (%)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>28.6</td>
<td>33.3</td>
<td>35</td>
<td>33.3</td>
<td>33.3</td>
<td>34.1</td>
</tr>
</tbody>
</table>

only one every time unit. Thus, the maximum number of buffered segments appears at time unit $n_{k-1} - 2$ as follows:

$$B(t) = B(k-3, n_{k-1} - 2) + B(k-2, n_{k-1} - 2) + B(k-1, n_{k-1} - 2)$$

$$\leq n_{k-2} - 1 + \left[ \frac{(n_{k-2} - 1) - n_{k-2} + 2}{2} \right],$$

from (5a) and (5b)

$$= n_{k-2} + \frac{n_{k-3}}{2} - 1$$

$$\leq \left[ \frac{n_{k-1}}{4} \right] + \frac{n_k}{2}.$$  \hspace{1cm} (C.2)

(3) For $n_{k-1} - 2 < t \leq n_k - 2$, the client has finished playing all the segments received from channels $C_1$ to $C_{k-3}$, and thus the client only buffers segments from channels $C_{k-2}$ to $C_k$ as follows:

$$B(t) = B(k-2, t) + B(k-1, t) + B(k, t)$$

$$\leq n_k - 2 - t + \left[ \frac{t - n_{k-1} + 2}{2} \right] + \left[ \frac{t - n_k + 2}{2} \right],$$

from (5a) and (5b)

$$\leq \left[ \frac{n_{k-1}}{4} \right] + \frac{n_k}{2}.$$  \hspace{1cm} (C.3)

(4) For $n_k - 1 < t \leq n_{k+1} - 2 - \lceil n_{k-1}/2 \rceil$, the client has finished playing all the segments received from channels $C_1$ to $C_{k-2}$ and only performs segment downloading on channels $C_{k-1}$ and $C_k$ as follows:

$$B(t) \leq B(k-1, t) + B(k, t)$$

$$\leq \left[ \frac{n_{k-1}}{2} \right] + \left[ \frac{t - n_{k-1} + 2}{2} \right],$$

from (5b)

$$\leq \left[ \frac{n_{k-1}}{4} \right] + \frac{n_k}{2},$$

due to $t \leq n_{k+1} - 2 - \left[ \frac{n_{k-1}}{2} \right].$  \hspace{1cm} (C.4)

(5) For $n_{k+1} - 2 - \lfloor n_{k-1}/2 \rfloor < t \leq n_{k+1} - 2$, similarly, the client merely downloads segments on channels $C_{k-1}$ and $C_k$ as follows:

$$B(t) \leq B(k-1, t) + B(k, t)$$

$$\leq n_{k+1} - 2 - t + \left[ \frac{t - n_{k-1}}{2} + \frac{2}{2} \right],$$

due to $n_{k+1} - 2 - \left[ \frac{n_{k-1}}{2} \right] < t$.  \hspace{1cm} (C.5)

(6) For $n_{k+1} - 2 < t$, the client simply performs data downloading on channel $C_k$, and thus $B(t) = B(k, t)$. From (5b), $B(t) = B(k, t) \leq \lfloor n_k/2 \rfloor \leq \lceil n_{k-1}/4 \rceil + \lfloor n_k/2 \rfloor$.

The proof is complete.

Acknowledgment

This work was financially supported by National Science Council, Taiwan under a research grant numbered NSC 101-2221-E-152-004.

References


