Adaptive Fast Data Broadcasting Scheme for Video-on-Demand Service

Li-Shen Juhn and Li-Ming Tseng

Abstract — As we support video-on-demand (VOD) service with batching schemes, the bandwidth requirement will be very large for a popular movie. For a hot video, fast data broadcasting scheme substantially reduce the bandwidth requirements as compared with batching schemes. However, the fast data broadcasting scheme needs to predict which movie is hot. If the prediction is not accurate, the allocated bandwidth will be wasted. This paper presents a new data broadcasting scheme for VOD service. For a movie, as it is popular, the new scheme will work like the fast data broadcasting scheme to save the communication bandwidth. If there is no request for the movie, the new scheme will not allocate bandwidth for the movie. Therefore, the bandwidth allocation for a movie is always efficient whether or not the movie is popular.

Index Terms — Bandwidth Utilization, Digital Video Broadcasting, Video-on-Demand.

I. INTRODUCTION

Consider an interactive video service where multimedia server can provide videos to users over a high speed network or digital satellite system [1,2]. The worst case of requirements to the server is that all users request different movies at the same time. If a server wants to accommodate the worst requirement case, the total bandwidth requirement will be very large. In order to prepare or reserve such a huge bandwidth, the cost is very high, but the bandwidth utilization is not efficient in a normal requirement case. Therefore, how to use limited bandwidth to service as many users’ requests as possible is a problem to be solved.

This work was supported by the National Science Council of the Republic of China under the grant NSC-87-2213-E-008-029.

The authors are with the Distributed System Laboratory, Department of Computer Science and Information Engineering, National Central University, Chungli, Taiwan 32054, R.O.C. E-mail: juhn@dslab.csie.ncu.edu.tw, tsenglm@csie.nchu.edu.tw.

In general, many users may request the same movie during a time interval. The batching schemes consider to delay the users’ requests for a certain amount of time so that users for the same movie within a time interval can share a common video stream to save bandwidth [3,4]. However, the batching schemes still need huge bandwidth for a popular movie. For example, suppose there is a popular movie which length is 120 minutes. If the time slot is 10 minutes and there is a request every time slot for this movie, we need to allocate 12 video channels for the movie.

Suppose the set-top box at the client end can buffer portions of the playing video on disk. For a popular movie, the pyramid [5], harmonic [6], staircase [7], and fast [8] broadcasting schemes substantially reduce the bandwidth requirements as compared with the batching schemes. For the above example, using the fast broadcasting scheme in [8], we only need to allocate 4 video channels as shown in Fig.1. However, the broadcasting schemes need to predict how hot the movie will be. If we allocate 4 video channels to broadcast a movie every 8 minutes with the fast broadcasting scheme but there is no customer to see the movie, we will waste the allocated bandwidth.

![Fig.1: An example of fast broadcasting scheme](image-url)

In this paper, we present an adaptive fast broadcasting scheme for VOD service. The bandwidth requirement for a movie depends on the users’ requests for the movie. If the movie is popular, the bandwidth requirement of the new scheme is the same as the fast broadcasting in [8]. However, if there is no request for the movie, the new scheme will not allocate any bandwidth for this movie. The bandwidth allocation for a movie is always efficient whether or not the movie is hot. We need not to do the prediction.
II. ADAPTIVE FAST DATA BROADCASTING AND RECEIVING SCHEME

Suppose there is a movie with length \( D \) (e.g., 120 minutes). The consumption rate of the movie is \( b \) (e.g., 10 Mbps). The size of the movie is, thus, \( S = D \cdot b \) (e.g., 9 Gbytes).

Divide the movie every \( d \) minutes (e.g., \( d = 8 \) minutes). Suppose the movie is divided into \( N \) segments. The last segment may contain some blank data. Therefore,

\[
N = \left\lfloor \frac{D}{d} \right\rfloor. \quad (\text{e.g., } N = 15)
\]  

(1)

Suppose \( S_i \) is the \( i \)th segment of the movie. The concatenation (\( \bigcirc \)) of all the segments, in the order of increasing segment numbers, constitutes the whole movie: \( S = S_1 \bigcirc S_2 \bigcirc \ldots \bigcirc S_N \).

Let \( S_i \) belongs to the data group \( G_p \). Where \( p = \left\lfloor \log_2 i \right\rfloor \). \hspace{1cm} (2)

Hence, \( G_0 = \{ S_1 \} \), \( G_1 = \{ S_2, S_3 \} \), \( G_2 = \{ S_4, S_5, S_6, S_7 \} \), ..., \( G_N = \{ S_{2^N}, \ldots, S_N \} \). Where \( q = 2^i \) and \( K = \left\lfloor \log_2 N \right\rfloor \). \hspace{1cm} (3)

Let \( C_0, \ldots, C_K \) represent the \( K + 1 \) possible allocated video channels for this movie. Initially, the \( K + 1 \) video channels are all empty (not allocated).

Let \( h \) represents the highest channel number that can be assigned for a new allocated video channel. The initial value of \( h \) is \( K \).

On the server side, the adaptive fast broadcasting scheme services the movie every \( d \) minutes in the following way:

1. Release the data segments that have been sent in the previous time interval from each allocated video channel. If there is no more data to be sent within the channel, release the video channel (suppose it is \( C_r \)). If the channel number (\( r \)) of the released video channel is greater than \( h \), let \( h = r \).
2. If there is no request for this movie during the last time interval, exit the procedure. Otherwise, do the next step.
3. If there is no free video channel, reject the request(s). Otherwise, according to the admission control policy, allocate a new video channel and put the rest data segment(s) of the movie into the allocated video channel(s) in the following way:

   (1) Assign \( C_h \) for the new allocated video channel. Let \( C_h = C_h \) and update the \( h \) to the highest channel number that still empty.

   (Note: at this moment, \( C_0, C_{a1}, \ldots, C_K \) should be allocated.)

(2) According to Table 1, put the rest data segment(s) in sequence into \( C_a, C_{a1}, \ldots, C_K \).

<table>
<thead>
<tr>
<th>Data segment belongs to</th>
<th>Put the data into</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G_0, G_{a1}, \ldots, G_a )</td>
<td>( C_a )</td>
</tr>
<tr>
<td>( G_{a1} )</td>
<td>( C_{a+1} )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( G_K )</td>
<td>( C_K )</td>
</tr>
</tbody>
</table>

Where the rest data segments of the movie are those data segments which are not in the allocated video channels currently.

Referring to Fig.2, suppose we divide a movie into 15 segments and there is a request every time interval for this movie. At the beginning \( t_0 \), we will allocate a video channel \( C_3 \) and put all of the 15 data segments into the channel. At \( t_0 + d \), the first data segment \( S_1 \) is released from \( C_3 \). We need to allocate another channel \( C_2 \) and put \( S_1 \) into \( C_2 \), because the \( S_1 \) is already not in the allocated video channel \( C_3 \). In this case, we will allocate 4 video channels to service this movie at \( t_0 + 6d \). After \( t_0 + 6d \), if there is still at least a request every time interval for this movie, it will work like the original fast broadcasting scheme as Fig.1. In this case, using the adaptive fast broadcasting scheme, we only need to allocate 4 video channels. If we use the conventional batching schemes, we need to allocate 15 video channels.

For a given movie, if there is no request during some interval, the bandwidth requirement will reduce. Fig.3 is an example. In the example, we also need to allocate 4 video channels at \( t_0 + 6d \). However, there is no request during \([t_0 + 6d, t_0 + 7d]\). We will release channel \( C_0 \) at \( t_0 + 7d \). In this example, if we use the original fast broadcasting scheme, we will always allocate 4 video channels.

At the client end, suppose there is plenty of disk space to buffer portions of the playing video. For watching a movie, the following steps are involved:

1. Request the movie and wait to get the channels’ assignment for the movie.
2. According to the channels’ assignment and the number of data segments in each allocated channel, begin to download the first data segment \( \{S_i\} \) of the required movie at its first occurrence and to download other related data segments concurrently.
3. Right after we begin to download the data segments, we can start to consume the movie with its consumption rate in the order of $S_i \cdot S_{i+1} \cdot \ldots \cdot S_N$.

![Diagram](image)

Fig. 3: Another example of the adaptive fast data broadcasting scheme (There is no request in some interval).

III. ANALYSIS AND COMPARISON

A. Workable Verification

Referring to Fig. 1, in the original fast broadcasting scheme, the data segment(s) of $G_i$ are periodically broadcasted in $C_i$, where $i = 0, \ldots, K$. At any starting point of a time interval, if we concurrently download the data segments from all of the video channel(s) and start to see the movie, we can receive any data segment before (includes in time) we need to consume the data segment at the client end. The reason is that we will consume all of the $2^{i+1}-1$ data segments within
\{C_0, \ldots, C_{n-1}\} \text{ before we start to consume the first data segment of } C_i, \text{ but there are at most } 2^i \text{ data segments in the } C_i.

Using the adaptive fast data broadcasting and receiving scheme, we can also get any data segment of a movie before we need to consume the data segment. The reason is that at any step we will allocate a video channel } C_a \text{ and place the rest data segment(s) in sequence according to Table 1.

1. At this moment, we can receive all of the data segments that are placed in } C_a \text{ before we need to consume them.

2. Before the } C_a \text{ is released, if } C_a \text{ is allocated continuously, only the data segments which belong to } G_a \text{ can be placed again into this channel. There are at most } 2^a \text{ data segments in the } G_a. \text{ The condition is similar to the original fast broadcasting scheme. Hence, we can still receive any data segment in } C_a \text{ before we need to consume the data segment.

B. Admission Control Policies

Suppose there are different requirements but there is only one free video channel. We need to decide which video is best to schedule at that particular moment in time. There has been a considerable amount of research devoted to scheduling policies for the batching technique [3,4]. For example, the first-come-first-served (FCFS) policy uses viewer's waiting time as the weighting factor to decide which one will be served first. The weighting factor of the maximum queue length (MQL) policy is the number of viewers waiting for the video. The video with the maximum number of waiting requests will be served first. The maximum factored queue length (MFQ) policy uses both the waiting time and queue length as the weighting factors [4]. We can also apply all of these policies to the adaptive fast broadcasting scheme for admission control. However, all of these admission control policies can not reduce the bandwidth requirement for a popular movie.

IV. CONCLUSION

Currently, the disk space and computation capability of a personal computer are very large and powerful. It seems achievable to buffer an entire playing video at the client end. However, in the near future, the communication bandwidth seems still not adequate to support an ideal VOD service over a wide area.

Although it is not feasible to support any movie at any time, it seems possible to support some videos with a specified waiting time. In current video rental stores, we find that most of the users are interested in the same few popular movies at about the same time. If we offer a near VOD service at a metropolitan area, we believe that most of the viewers will also want some popular videos during a period of time.

For a popular movie, suppose the client can buffer portions of the playing video on disk, the fast broadcasting scheme greatly reduces the bandwidth requirement, but we need to predict which movie is hot. If the prediction is not accurate, we will waste the allocated bandwidth. This paper presents an adaptive fast broadcasting scheme, which needs not to do the prediction. For a given movie, if there is a request every time interval for this movie, the new scheme will work like the original fast broadcasting scheme. If the access rate of the movie is reduced, the bandwidth requirement is also reduced. The bandwidth utilization is always efficient whether or not the movie is popular.

REFERENCES


