

# An optimal broadcasting protocol for mobile video-on-demand

Regant Y.S. Hung      H.F. Ting\*

Department of Computer Science  
The University of Hong Kong  
Pokfulam, Hong Kong  
Email: {yshung, hfting}@cs.hku.hk

## Abstract

The advance of wireless and mobile technology introduces a new type of Video-on-Demand (VOD) systems, namely the mobile VOD systems, that provide VOD services to mobile clients. It is a challenge to design broadcasting protocols for such systems because of the following special requirements: (1) fixed maximum bandwidth requirement: the maximum bandwidth required for broadcasting a video should be fixed and independent of the number of requests, (2) load adaptivity: the total bandwidth should be dependent on the number of requests; the fewer the requests the smaller the total bandwidth usage, and (3) clients sensitivity: the system should be able to support clients with a wide range of heterogeneous capabilities. In the literature, there are some partial solutions that give protocols meeting one or two of the above requirements. In this paper, we give the first protocol that meets all of the three requirements. The performance of our protocol is optimal up to a small constant factor.

*Keywords:* wireless network, broadcasting, video-on-demand.

## 1 Introduction

Due to the increasing popularity of wireless networks, mobile Video-on-Demand systems, which provide VOD services to mobile clients, have found many practical applications. For example, airlines can now provide VOD services in airport lounges to entertain the waiting passengers on their laptops or PDAs. Universities can install mobile VOD systems that allow students to watch important video lectures anywhere anytime on campus. As pointed out by Tran *et al.* (2004), designing broadcasting protocols for mobile VOD systems is different from designing those for the traditional ones. In particular, we have the following three special requirements for mobile VOD systems.

**Fixed maximum bandwidth:** A wireless network can only support a limited amount of bandwidth. For example, the IEEE 802.11g standard provides a maximum bandwidth of 54Mbps. Therefore, a video server enabled with 802.11g cannot deliver more than thirty-six 1.5Mbps MPEG1 video streams simultaneously. In other words,

the server has only thirty-six channels, and if we want to broadcast six popular movies at the airport waiting lounge, we can use only six channels for each movie. It is not an easy task to broadcast a movie on six channels such that a large number of clients, say 100, arriving and hooking-up to the system at arbitrary times, can watch the movie from the beginning with near-zero delay.

**Load adaptivity.** The load of a VOD system is usually distributed unevenly; it is heavy only over a short period of time. For instance, in our airport lounge example, the system will have a heavy load only during one or two hours before a flight departure. Therefore, the system should be able to adapt to different loads and make necessary adjustment to the broadcasting schedule so as to minimize the total bandwidth usage. Load adaptivity is even more important to mobile VOD system because of its energy consumption. Since the coverage of wireless transmission is limited, we often need multiple hosts to cover a large enough service area; thus, a significant amount of energy for the intermediate mobile hosts is consumed. The system should use smaller total bandwidth when the load is light so as to save energy.

**Client sensitivity.** In a mobile VOD system, clients use their own equipments to watch the video broadcasting. These equipments fall in a wide spectrum of heterogeneous capabilities, ranging from powerful laptops to primitive PDAs. The system should be able to serve even the most primitive clients. On the other hand, it should fully exploit the resources that a client is capable of or willing to use for watching the video in order to provide the best possible quality of services. For example, it should be sensitive to the buffer size such that a client with larger buffer should be able to watch the video with small delay.

Broadcasting techniques such as *video skimming* (Eager, Vernon & Zahorjan 2000), *streams merging* (Chan, Lam, Ting & Wong 2002, Chan, Lam, Ting & Wong 2005, Bar-Noy & Ladner 2003) and *piggy-backing* (Golubchik, Lui & Muntz 1996), which are successful for traditional VOD systems, are not applicable to mobile VOD systems because their maximum bandwidth requirements are not fixed. For example, in a streams merging system, the maximum bandwidth required over some time-span depends on the number of requests arriving in that time-span (if there are  $r$  such requests, the maximum bandwidth is  $O(\log r)$  channels). An obvious approach to meet the maximum bandwidth requirement is to divide the time-span of a video of  $D$  minutes into intervals of  $\delta$  minutes and broadcast the video every  $\delta$  minutes. This scheme uses  $D/\delta$  channels and guarantees a maximum waiting time of  $\delta$  minutes. Juhn and

\*This research was supported in part by Hong Kong RGC Grant HKU-7045/02E  
Copyright ©2007, Australian Computer Society, Inc. This paper appeared at Computing: The Australasian Theory Symposium (CATS2007), Ballarat, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 65. Joachim Gudmundsson and Barry Jay, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Tseng (1997) showed that the maximum bandwidth requirement can be reduced exponentially provided that clients have enough memory to buffer about half of the video content. More precisely, they proposed the *Harmonic broadcasting protocol*, which assumes that every client has a buffer of size at least  $\frac{2}{5}Db$  where  $b$  is the bit rate. They proved that their protocol uses only a maximum of  $\ln(D/\delta)$  channels to guarantee a maximum waiting time of  $\delta$  minutes. Equivalently, if we have  $\beta$  channels to broadcast a video, then Harmonic broadcasting guarantees a maximum waiting time of  $\delta = D/e^\beta$  (see Table 1).

Although Harmonic broadcasting has fixed maximum bandwidth, it is neither load adaptive nor client sensitive. In view of this deficiency, Juhn and Tseng (1998) later proposed a client-sensitive protocol called *Fast broadcasting*, which supports clients with different buffer sizes, though the maximum waiting times depend on how much buffer the clients can use to store parts of the video. To be precise, let  $D$  be the length of the video,  $b$  is the broadcast bit rate, and  $\beta$  be the maximum bandwidth we can use. For each  $0 \leq j \leq \beta$ , define  $\delta_j = \frac{2^j}{2^{\beta-1}}$ . As shown in Table 1, Fast broadcasting guarantees a maximum waiting time of  $\delta_j D$  provided that a client has enough buffer to store about a fraction of  $\frac{1}{2} - \delta_j$  of the total  $Db$  bits of video. However, Fast broadcasting is not adaptive to load; its total bandwidth is independent of the number of requests. In a time-span of  $T$  minutes, its total bandwidth usage is  $T\beta$ .

Recently, Biedl *et al.* (2003) proposed the *Adaptive pyramid broadcasting* protocol, which is load adaptive; when there are  $v$  requests arriving in a timespan of  $T$  minutes, the total bandwidth used by the protocol to serve these  $v$  requests is at most  $T \min\{\lg \frac{Dv}{T}, \beta\}$ . Note that the total bandwidth usage is reduced significantly when  $v$  is small. However, Adaptive pyramid broadcasting is not client sensitive; it requires all clients to have enough memory to store about half of the video (see Table 1).

In this paper, we propose the first broadcasting protocol that meets all the three requirements for mobile VOD. It is client sensitive and the tradeoff between maximum waiting time and buffer usages matches that of the Fast broadcasting; if a client has a buffer of size about  $(\frac{1}{2} - \delta_j)Db$  bits, then the protocol guarantees a maximum waiting time of  $\delta_j D = \frac{2^j}{2^{\beta-1}}D$  for this client. If we call the ratio between maximum waiting time and video length  $D$  the *delay ratio* and that between buffer size and video size  $Db$  the *buffer ratio*, Figure 1(a) shows the tradeoff between delay ratio and buffer ratio guaranteed by our protocol.

Our protocol is also load adaptive. If there are  $v$  requests arriving during a time-span of  $T$ , then the total bandwidth used by our protocol for serving these requests is

$$T \min\{\lg \frac{Dv}{T}, \beta\}.$$

Note that this total bandwidth usage matches that of Adaptive pyramid broadcasting, and it has been proved by Biedl *et al.* (2003) that this performance is optimal up to a multiplicative factor of 1.44. Figure 1(b) shows the total bandwidth usage of our algorithm in terms of the number of viewing requests when  $\beta = 4$  and  $T = D = 16$ .

**Organization.** The rest of the paper is organized as follows. We describe the algorithm in Section 2. We prove the correctness of our algorithm in Section 3. We finally give the maximum waiting time of clients guaranteed and estimate the total bandwidth usage of our algorithm in Section 4.

## 2 Algorithm

In this section, we will give the algorithm and examples demonstrating how the algorithm works. Suppose the total length of the video is  $\mathcal{D}$  (e.g., 120 minutes). The consumption rate of the video is  $b$  (e.g., 500Kbps). Therefore, the size of the video is  $\mathcal{D}b$  (i.e.  $(120 \times 60) \times (500 \div 8) \approx 440\text{MB}$ ). Suppose we have  $\beta$  broadcasting channels where each of them has the bandwidth of  $b$ . Therefore, the time taken for a channel to broadcast a video segment is the same as the length of the segment.

The whole video is divided into  $\beta$  segments  $s_0, s_1, \dots, s_{\beta-1}$  so that the concatenation of the segments  $s_0 \cdot s_1 \cdot \dots \cdot s_{\beta-1}$  will form the whole video. Their lengths are different. For any  $1 \leq i \leq \beta - 1$ ,  $|s_i| = 2^i |s_0|$  where  $|s_i|$  denotes the length of the segment  $s_i$ . Set  $|s_0| = \mathcal{D}/(2^\beta - 1)$  such that  $\sum_{i=0}^{\beta-1} |s_i| = \mathcal{D}$ . Let  $\delta = \mathcal{D}/(2^\beta - 1)$  and  $\delta_j = 2^j \delta$ . Note that  $\delta_0 = \delta$  and  $|s_j| = \delta_j$  for  $j = 0, 1, \dots, \beta - 1$ . Recall that we have  $\beta$  channels. We label the  $\beta$  channels as  $c_0, c_1, \dots, c_{\beta-1}$ .

Recall that our algorithm adapts to the clients with different buffer sizes. We classify the clients into different classes according to their buffer sizes as follows. Let  $B_j$  denote a set of clients in which every client has buffer size in the range  $[(\delta_{\beta-1} - \delta_j)b, (\delta_{\beta-1} - \delta_{j-1})b)$  for  $1 \leq j \leq \beta - 1$ .  $B_0$  denotes the clients with the buffer no less than  $(\delta_{\beta-1} - \delta)b$ . Therefore,  $B_0$  is the class of clients who have enough buffer. The clients with no buffer belong to  $B_{\beta-1}$ .

If there is no client comes along, then the video server will not broadcast any segment to save the bandwidth. Once a client of class  $B_j$  arrives at some time  $t$  for any integer  $0 \leq j \leq \beta - 1$ , then

**Server side** Let  $i$  be some integer such that  $t \in ((i-1)\delta_j + \delta, i\delta_j + \delta]$ . The server will broadcast the segments in the following way:

**Segments  $s_0, \dots, s_{j-1}$ :** For any  $0 \leq k \leq j - 1$ , start to broadcast  $s_k$  through  $c_k$  at time  $i\delta_j + \delta_k$ .

**Segments  $s_j, \dots, s_{\beta-1}$ :** For any  $j \leq k \leq \beta - 1$ , start to broadcast  $s_k$  through  $c_k$  at time  $a\delta_k$  for some integer  $a$  such that  $(a-1)\delta_k < (i+1)\delta_j \leq a\delta_k$ .

**Client side** Start to download  $s_0$  at time  $i\delta_j + \delta$ . Start to play the video at  $i\delta_j + \delta$ . The client downloads the segments in the following way:

**Segments  $s_0, \dots, s_{j-1}$ :** For any  $0 \leq k \leq j - 1$ , download  $s_k$  through  $c_k$  at time  $i\delta_j + \delta_k$ .

**Segments  $s_j, \dots, s_{\beta-1}$ :** For any  $j \leq k \leq \beta - 1$ , download  $s_k$  through  $c_k$  at time  $a\delta_k$  for some integer  $a$  such that  $(a-1)\delta_k < (i+1)\delta_j \leq a\delta_k$ .

The maximum waiting time depends on how large buffer the client has. The maximum waiting time of clients of class  $B_j$  is equal to  $\delta_j = 2^j \mathcal{D}/(2^\beta - 1)$ . The examples demonstrating how our algorithm works are shown in Figure 2. In our examples, there are four channels and the video is divided in to four segments. In Example 1, there are 16 clients of class  $B_0$  arriving evenly during  $[0, 15\delta]$ . There are 16 clients of class  $B_3$  arriving evenly during  $[0, 15\delta]$  in Example 2. In Example 3, there are four clients of four different classes arriving at time  $3.5\delta$ . In the figure, we label the segments with the numbers in the range from zero to three in order to show that which segment will be downloaded by which client. For instance, the segment  $s_0$  broadcasted by  $c_0$  at time  $4\delta$  has label "0". This means that this segment will be downloaded by

	Total bandwidth over a time-span of $D$	Maximum waiting time	Buffer requirement
Harmonic Broadcast	$D\beta$	$\frac{1}{e^\beta} D$	$\approx \frac{2}{5} Db$ .
Fast Broadcast	$D\beta$	$\frac{2^j}{2^\beta - 1} D$	$(\frac{2^{\beta-1} - 2^j}{2^\beta - 1}) Db$
Adaptivity Pyramid	$\approx D \min\{\lg v, \beta\}$	$\frac{1}{2^{\beta-1}} D$	$\approx \frac{1}{2} Db$
Our protocol	$\approx D \min\{\lg v, \beta\}$	$\frac{2^j}{2^\beta - 1} D$	$(\frac{2^{\beta-1} - 2^j}{2^\beta - 1}) Db$

Table 1: Performance of different protocols.

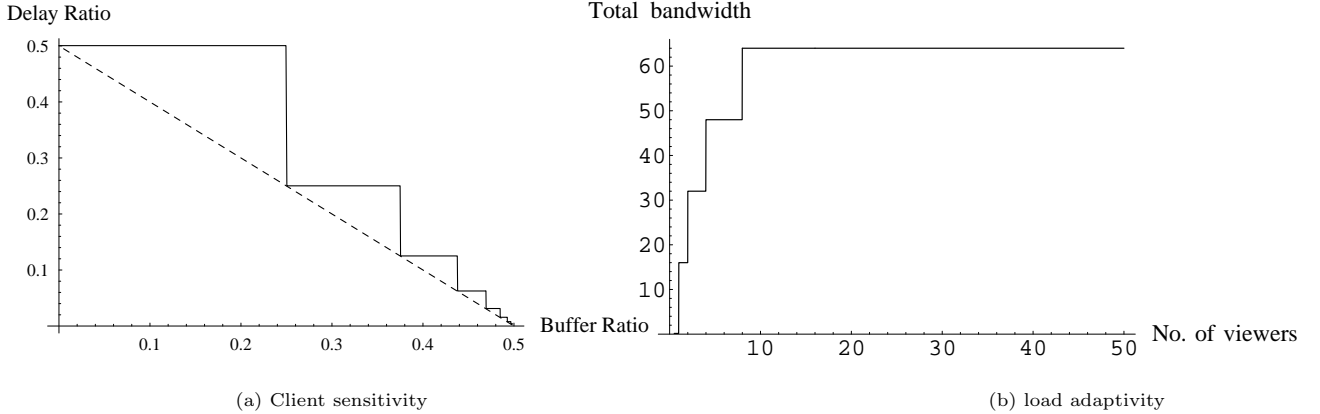


Figure 1: Performance of our algorithm

the client of the class  $B_0$  only. Similarly, the segment  $s_3$  broadcasted by  $c_3$  at time  $8\delta$  will be downloaded by the clients of the classes  $B_0, B_1$  and  $B_2$ . In the last example, there are three clients in 3 different classes arriving in different times. One client in  $B_3$  arrives at  $\delta$ . One client in  $B_1$  arrives at  $5.7\delta$ . One client in  $B_0$  arrives at  $7.8\delta$ .

### 3 Proof of correctness

In this section, we will prove the correctness of our algorithm. We need to prove the following.

1. Each channel will broadcast one segment at a time. That is, no two parts of the same segment will be broadcast in parallel.
2. Clients starts downloading any segment no later than the start of playing of this segment.
3. Clients will have enough buffer for buffering the video.

These are proved by the following three theorems.

**Theorem 1.** *No two parts of the same segment will be broadcast in parallel.*

We first give the following lemma that is useful for proving the theorem.

**Lemma 2.** *For any integer  $0 \leq i \leq \beta - 1$ ,  $s_i$  never starts being broadcast at time  $t \neq j\delta_i$  for any integer  $j \geq 0$ .*

*Proof.* For any client in  $B_k$  where  $k > i$ , the server broadcasts  $s_i$  at some time  $m\delta_k + \delta_i$  for some integer  $m$ . Since  $k > i$ ,  $\delta_i$  is a factor of  $\delta_k$  and thus,  $m\delta_k + \delta_i = m'\delta_i + \delta_i = (m' + 1)\delta_i$  where  $m' = m\delta_k/\delta_i$ . Since  $m'$  is an integer, the server will start to broadcast  $s_i$  for

any client in  $B_k$  only at time  $j\delta_i$  for some integer  $j$  and for any integer  $i < k$ . For the clients belonging to  $B_{k'}$  where  $0 \leq k' \leq i$ , the server will broadcast  $s_{k'}$  for these clients at  $a\delta_{k'}$  for some integer  $a$ . Therefore, for any integer  $0 \leq i \leq \beta - 1$ , the server will not start to broadcast  $s_i$  at time  $t \neq j\delta_i$  for any integer  $j \geq 0$ .  $\square$

*Proof of theorem 1.* By Lemma 2,  $\delta_i, 2\delta_i, \dots$  are the only possible time instants that  $s_i$  starts being broadcast. Together with the fact that  $|s_i| = \delta_i$ , no two parts of a segment will be broadcast in parallel.  $\square$

Let us give the following observation before proving the following two theorems.

**Observation 3.** *Consider any client  $v$  of class  $B_j$  for some integer  $j$ . For the segments  $s_0, s_1, \dots, s_{j-1}$ ,  $v$  will download these segments one by one and play it at the same time.*

*Proof.* Consider any client  $v$  in  $B_j$  arriving at  $((i - 1)\delta_j + \delta, i\delta_j + \delta]$  for any integer  $i \geq 0$ . Since  $v$  starts to play the video at  $i\delta_j + \delta$  and  $v$  needs to play  $s_i$  after playing segments  $s_0, s_1, \dots, s_{i-1}$ ,  $v$  must start to download  $s_i$  no later than

$$\begin{aligned}
& i\delta_j + \delta + |s_0| + |s_1| + \dots + |s_{i-1}| \\
&= i\delta_j + \delta + \delta_0 + \delta_1 + \dots + \delta_{i-1} \\
&= i\delta_j + \delta_k
\end{aligned}$$

Recall that  $v$  will download  $s_k$  through  $c_k$  at time  $i\delta_j + \delta_k$  for any  $0 \leq k \leq j - 1$  according to the algorithm. Thus,  $v$  will download these segments one by one and play it at the same time.  $\square$

This observation helps to prove the following two theorems.

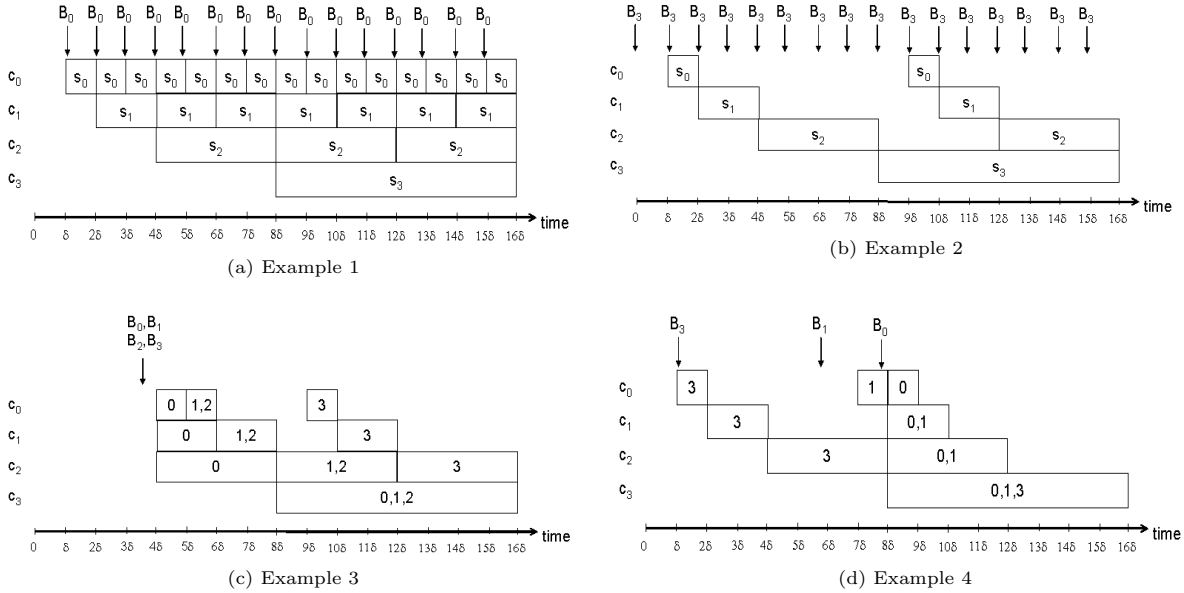


Figure 2: Demonstration of our algorithm

**Theorem 4.** Any client  $v$  in  $B_j$  can start to download any segment no later than the start of playing this segment for any integer  $0 \leq j \leq \beta - 1$ .

*Proof.* By Observation 3, for any  $1 \leq k \leq j - 1$ ,  $v$  can start to download  $s_k$  no later than the start of playing  $s_k$ . We now prove that for any  $j \leq k \leq \beta - 1$ ,  $v$  can start downloading  $s_k$  no later than the start of playing it. We prove this by contradiction. Recall that for any integer  $j \leq k \leq \beta - 1$ , the client will download  $s_k$  at  $a\delta_k$  such that

$$(a - 1)\delta_k < (i + 1)\delta_j \quad (1)$$

and  $(i + 1)\delta_j \leq a\delta_k$ . As stated,  $v$  will play  $s_k$  at  $i\delta_j + \delta_k$ . Assume that the client does not download  $s_k$  during  $[(i + 1)\delta_j, i\delta_j + \delta_k]$  such that  $v$  cannot download  $s_k$  no later than playing it. This implies that

$$i\delta_j + \delta_k < a\delta_k,$$

or equivalently,

$$i\delta_j < (a - 1)\delta_k \quad (2)$$

Combining Inequalities 1 and 2, we have

$$i\delta_j < (a - 1)\delta_k < (i + 1)\delta_j$$

and thus,

$$i < (a - 1)\frac{\delta_k}{\delta_j} < i + 1.$$

Note that  $k \geq j$  and thus,  $\delta_j$  is a factor of  $\delta_k$ . Thus,  $\frac{\delta_k}{\delta_j}$  is an integer. By definition,  $a$  is also an integer and thus,  $(a - 1)\frac{\delta_k}{\delta_j}$  is also an integer. Again, by definition,  $i$  is an integer. However, it is impossible that there exist another integer  $((a - 1)\frac{\delta_k}{\delta_j})$  in between  $i$  and  $i + 1$ . This leads to contradiction. Hence,  $v$  will download  $s_k$  in  $[(i + 1)\delta_j, i\delta_j + \delta_k]$  for any integer  $j \leq k \leq \beta - 1$ .  $\square$

**Theorem 5.** The buffer usage of the any client  $v$  in  $B_j$  is at most  $(\delta_{\beta-1} - \delta_j)b$  at any time during accessing the video for any integer  $0 \leq j \leq \beta - 1$ .

*Proof.* By Observation 3, no buffer is needed while downloading  $s_0, s_1, \dots, s_{j-1}$  since  $v$  will download and play those segments simultaneously. The worst case is that they start to download the segments  $s_j, s_{j+1}, \dots, s_{\beta-1}$  in parallel. Since when the clients download those segments, they will play the segment at the same time. Therefore the worst case buffer usage for  $v$  equals the total size of the segments  $s_j, s_{j+1}, \dots, s_{\beta-1}$  minus the amount of data played during downloading these segments, that is,

$$\begin{aligned} & (|s_j|b + |s_{j+1}|b + \dots + |s_{\beta-1}|b) - \delta_{\beta-1}b \\ &= \delta_j b + \delta_{j+1}b + \dots + \delta_{\beta-2}b \\ &= \delta_{\beta-1}b - \delta_j b \end{aligned}$$

The theorem follows.  $\square$

## 4 Performance analysis

### 4.1 Maximum waiting time against the size of buffer available

Recall that, for any integer  $0 \leq j \leq \beta - 1$ , a client of class  $B_j$  arriving at some time  $t \in I = ((i - 1)\delta_j + \delta, i\delta_j + \delta]$  for some integer  $i$  will play the video at  $i\delta_j + \delta$ . Note that the length of  $I$  is  $\delta_j$  and thus, the maximum waiting time for clients of class  $B_j$  is at most  $\delta_j = 2^j \delta = 2^j \mathcal{D} / (2^\beta - 1) \approx \mathcal{D} / 2^{\beta-j}$ . We give Table 2 that shows a brief and clearer picture of our results.

### 4.2 Total bandwidth usage

We now show that the total bandwidth usage of our algorithm is optimal asymptotically. That is, given any request sequence, there does not exist any other algorithm that can consume less bandwidth in total than our algorithm asymptotically. In order to estimate the total bandwidth usage of our algorithm, we adopt the model used by Biedl *et al.* (2003). Since our algorithm is satisfying the two conditions stated in Lemma 1 of their works (Biedl, Demaine, Golynski, Horton, López-Ortiz, Poirier, & Quimper 2003), our algorithm is optimal in total bandwidth usage. Therefore, we will use the same technique as shown in that paper when we prove the optimality of our algorithm.

Size of buffer dedicated to the video	Maximum waiting time
0 (no buffer)	$\approx 1/2$ of the size of video
$\gg 1/4$ of size of video	$\approx 1/4$ of size of video
$\gg 1/2$ of size of video	$D/(2^\beta - 1)$

Table 2: Maximum waiting time for clients with different buffer size

First of all, let us describe the model that is used to estimate the total bandwidth usage of any algorithm. Let  $\delta$  be the maximum waiting time the clients need to wait before playing the video. With the same settings of the paper of Biedl *et al.* (2003), we divide the entire broadcasting duration into timespans of  $\mathcal{T} = m\delta$  for some integer  $m$ . Suppose the length of video equals  $n\delta$ . Note that some segments requested by the clients arriving in  $[0, \mathcal{T})$  may be broadcast in  $[\mathcal{T}, 2\mathcal{T})$ . We ignore the bandwidth usage for such segments. We only count those segments broadcast during the current timespan. Provided that  $\mathcal{T}$  is large enough, the ignored bandwidth usage is negligible compared with the total bandwidth usage. Let  $v$  denote the number of clients arriving in some timespan  $I$  of length  $\mathcal{T}$ . Just like the work of Biedl *et al.* (2003), when we estimate the total bandwidth usage, we will not include the consumption rate and the maximum waiting time for the calculation of the total bandwidth usage for simplicity. For example, if the server broadcasts some segment on a channel for  $\alpha\delta$  time units, we say its total bandwidth usage is  $\alpha$ . Biedl *et al.* (2003) also gave the lower bound of total bandwidth usage for any on-line algorithm.

**Theorem 6.** *Consider any online algorithm that ensures the maximum waiting time equals  $\delta$  and the video has the length of  $n\delta$ . If there are  $v$  clients arriving at equally spaced times in  $I$ , the total bandwidth usage of any online algorithm is no less than  $m \ln(\min\{(n+1)v/m, n\}) + O(m)$  in  $I$  where the video has the length of  $n\delta$ .*

*Proof.* By Theorems 1 and 4 in the work of Biedl *et al.* (2003).  $\square$

Here, our algorithm divides the video into segments where the smallest segment  $s_0$  has the length of  $\delta$ . Therefore, the maximum waiting time of the clients of class  $B_0$  is  $\delta$  time units. We divide the video into  $k = \lceil \log(n+1) \rceil$  segments such that the  $i$ -th segment  $s_i$  has size  $2^i\delta$  for any positive integer  $0 \leq i \leq k-1$ . We will have  $k$  channels. Channel  $i$  will broadcast  $s_i$  for any positive integer  $0 \leq i \leq k-1$ . We now prove that the total bandwidth usage of our algorithm is within a factor of optimal.

**Theorem 7.** *If there are  $v$  clients arriving in some timespan  $I$ , the total bandwidth usage of our algorithm is at most  $m \min\{\lceil \log(n+1) \rceil - \log(m/v) + 1, \lceil \log(n+1) \rceil\}$ .*

*Proof.* If  $v \geq m$ , the total bandwidth usage is no more than  $mk = m\lceil \log(n+1) \rceil$  since there are only  $k$  channels in our algorithm and the length of  $I$  is  $m$  (recall that  $\delta$  and the consumption rate are not involved in the estimation of the total bandwidth usage).

If  $v < m$ , we divide the channels into two batches: a batch containing  $c_0, c_1, \dots, c_{\log(m/v)-1}$  that broadcasts the segments  $s_0, s_1, \dots, s_{\log(m/v)-1}$  and the other batch containing the rest of the channels. We then estimate the total bandwidth usage of our algorithm by estimating the total bandwidth usage for each batch of channels. In our algorithm, the server will broadcast every segment at most once for every client. Therefore, the total bandwidth usage for

broadcasting the segments  $s_0, s_1, \dots, s_{\log(m/v)-1}$  is  $v \sum_{j=0}^{\log(m/v)-1} 2^j \leq m$ . On the other hand, since there is only one channel allocated to each segment and no two parts of a segment are broadcast in parallel, the total bandwidth usage for any channel in  $I$  is at most  $m$ . Thus, the total bandwidth usage for broadcasting the segments  $s_{\log(m/v)}, s_{\log(m/v)+1}, \dots, s_{k-1}$  in  $I$  is  $m(k - \log(m/v)) = m(\lceil \log(n+1) \rceil - \log(m/v))$ . As a result, the total bandwidth usage for our algorithm is at most  $m(\lceil \log(n+1) \rceil - \log(m/v) + 1)$  if  $v < m$ .

Thus, the total bandwidth usage of our algorithm is at most  $m \min\{\lceil \log(n+1) \rceil - \log(m/v) + 1, \lceil \log(n+1) \rceil\}$ .  $\square$

Recall that the unit of total bandwidth usage estimated above is the product of maximum waiting time and the consumption rate. Let  $\beta$  denote the number of channels used. If we multiply the the total bandwidth usage as shown above with maximum waiting time, we have the total bandwidth usage of  $T \min\{\lceil \log(n+1) \rceil - \log(m/v) + 1, \beta\}$  in the unit of consumption rate. If  $\log(n+1)$  is an integer, then  $T \min\{\lceil \log(n+1) \rceil - \log(m/v) + 1, \beta\} = T \min\{\log((n+1)v/m) + 1, \beta\} \approx T \min\{\log(\frac{Dv}{T}) + 1, \beta\}$ . Thus, we have the following corollary.

**Corollary 8.** *If  $\log(n+1)$  is an integer, then the total bandwidth usage of our algorithm is approximately equal to  $T \min\{\log(\frac{Dv}{T}) + 1, \beta\}$ .*

## 5 Conclusion

We have given in this paper the first broadcasting protocol that satisfies the three requirements for mobile VOD system, namely fixed maximum bandwidth, load adaptive and client sensitive. The total bandwidth required by our protocol is optimal up to a constant factor of at most 1.44. However, we believe that there is still room for improvement for the maximum waiting time. We note that when a client has enough buffer to store half of the video, our protocol guarantees a maximum waiting time of  $\frac{1}{2^\beta-1}D$ , while the Harmonic broadcast protocol guarantees a much smaller maximum waiting time, namely  $\frac{1}{e^\beta}D$ . It is an interesting problem to find a better client sensitive protocol that has maximum waiting time comparable to that of Harmonic broadcasting when clients have large enough buffer.

## References

- Bar-Noy, A. & Ladner, R. E., (2003), Competitive on-line stream merging algorithms for media-on-demand, *Journal of Algorithms*, 48(1), pp. 59–90.
- Biedl, T. C., Demaine, E. D., Golynski, A., Horton, J. D., López-Ortiz, A., Poirier, G. & Quimper, C., (2003), Optimal dynamic video-on-demand using adaptive broadcasting, *Proceedings of the 11th Annual European Symposium on Algorithms (ESA)*, pp. 90–101.

- Chan, W. T., Lam, T. W., Ting, H. F. & Wong, P. W. H., (2002), A unified analysis of hot video schedulers, *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, pp. 179–188.
- Chan, W. T., Lam, T. W., Ting, H. F., & Wong, P. W. H., (2005), On-line Stream Merging with Max Span and Min Coverage, *Theory of Computing Systems*, 38:461–479.
- Eager, D., Vernon, M. & Zahorjan, J., (2000), Bandwidth skimming: A technique for cost-effective video-on-demand, *Proc. Conf. on Multi. Comput. and Net. (MMCN)*, pp. 206–215.
- Golubchik, L., Lui, J. C. S., & Muntz, R. R., (1996), Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers, *Multimedia Systems*, Vol. 4, pp. 140–155.
- Juhn, L., & Tseng, L., (1997), Harmonic broadcasting for video-on-demand service, *IEEE Transactions on Broadcasting*, Vol. 43(3), pp. 268–271.
- Juhn, L., & Tseng, L., (1998), Fast data broadcasting and receiving scheme for popular video service, *IEEE Transactions on Broadcasting*, Vol. 44(1), pp. 100–105.
- Tran, D. A., Le, M., & Hua, K. A., (2004), MobiVoD: a video-on-demand system design for mobile ad hoc networks, *IEEE International Conference on Mobile Data Management*, pp. 212–223.