A Cost-Effective Interactive Broadcasting Protocol for Media Streaming

Shu-Zhe Weng Simon Sheu Jun-Yi Li
Computer Science, National Tsing Hua University
Hsinchu, Taiwan 30013, Republic of China
{sjwong, sheu, jylee}@cs.nthu.edu.tw

ABSTRACT

On-demand media streaming can offer users instantly access of videos. Many Video-on-Demand (VoD) broadcasting protocols for the popular videos have been shown to be efficient to reach this ideal. Unfortunately, these techniques were optimized only for normal playback. Interactive services, such as *fast-forward* or *rewind*, are left intact as hard options. To meet this challenge, we propose a novel broadcasting protocol for the clients with sufficient disk buffer to enjoy VCR-style services. These users only need to accumulate *three* broadcast streams, while the server merely patches the unavailable momentarily. With this unique feature, we minimize the demands on the communication and storage bandwidths of all clients. As such, cost-effective interactive VoD services can be easily realized over a large population of users. We evaluate the overhead of the proposed scheme analytically, and perform an intensive emulation over all the situations of interactive services. Performance study shows that our scheme can provide the comparable services only demanding much less client bandwidths.

Keywords: Media streaming, video-on-demand, VCR services, broadcasting protocols.

1. Introduction

Based on media streaming, VoD is an attractive network multimedia application that allows users to play back videos at any time. This objective is similar to its analog counterparts: pay-per-view (PPV) and videocassette rentals, which provide educational and entertaining media contents. To make the digital transition successful on the marketplace, VoD, the digital form of PPV, needs to overcome the challenge of reducing the average service cost. The common approach of existing techniques employs the multicast or broadcast facilities to maximize the extent of media streams sharing. The cost of server streams can be therefore amortized over many users. The result of lower per service cost renders a more economic solution than the conventional PPV.

There are basically two service models to magnify server streams sharing. The first model is *reactive* [3] in the sense that the service streams are set up on demand according to the user requests. The media server then intervenes the users requesting for the same media object to receive the on-going delivery of common portion of this object [12]. Avoiding the unnecessary transmission of data is thus achieved. For instance, User B starts the playback of the video two minutes after User A. By directing User B to also receive the data multicast to User A, the media server needs only to stream the first two minutes of the video, instead of the entire video, to User B. However, when the requests arrive too frequently beyond the server capacity, the server will not be able to fulfill the requests promptly. To address this problem, the media server of the second service model periodically delivers the video preceding the user requests [5]. Due to the anticipation of the future requests, this service model is referred to as *proactive* [12]. The service latency can be confined within the period of video delivery, no matter how many users are requesting the video.

The rationale behind both models to maximize streams sharing is to take the advantage of the characteristics of normal video playback. Each video

The authors are supported by the National Science Council, Project No. NSC 89-2213-E-007-125, Taiwan, R.O.C.

can be seen as the sequential concatenation of video segments, one after one. Users playing out the current segment will continue to play back the next segment at the same pace. As a result, the last video segment is most likely the common need of all users for the same video [5]. Using previous example, despite two-minute difference in start time, User A and B commonly demand later video segments. Consequently, when each segment is delivered on the shared streams to all demanding users just prior to its playback, service streams sharing can be optimized [3, 11].

Unfortunately, when users perform VCR-style operations, such as *fast-forward*, *forward jump* or the like, the anticipation of common segments needed among these interactive users become very difficult [3, 11]. Especially, most efficient broadcasting protocols try to deliver segments just in time for regular playback to conserve server bandwidth. An x times *fast-forward* operation will lead to x-1 times shortage of video data. To overcome this problem, Pâris recently proposed a scheme called *Interactive Pagoda Broadcasting* (IPB) protocol to support *forward jump* function by segment patching [12]. The idea, like the approach in [3], is to retain all the received video data locally in each client disk buffer, while transmitting missing video segments on demand. Through the disk buffer large enough to store the entire video, the service cost can be reduced by at least 50 percent. In this paper, we propose an alternative broadcasting protocol to achieve the comparable service cost savings, only utilizing constant communication bandwidth for each client. Unlike IPB, the proposed technique only requires each client to accumulate at most *three* broadcast streams. As a result, only a minimal disk storage bandwidth is required. With more economic client hardware requirements, cost-effective interactive VoD services can be easily realized over a large population of users.

The rest of this paper is organized as follows. Section 2 discusses the previous related work to make the paper self-contained. Section 3 presents the proposed broadcasting technique in further details. We perform the performance investigation in Section 4. Finally, Section 5 gives our concluding remarks and discusses future works.

2. Related Work

Broadcasting protocols for video-on-demand can be roughly classified into three groups. Protocols in the first group partition each video into fixed-size segments and use channels with different bandwidth to transmit them. Protocols in this category includes: Harmonic Broadcasting [7] and its variations: *Cautious Harmonic Broadcasting*, *Quasi Harmonic Broadcasting* [16], and *Polyharmonic Broadcasting* [15]. Protocols in second group partition the video into segments with increasing size, and use fixed bandwidth channels to transmit these segments. This category includes: *Pyramid Broadcasting* [20], *Fast Broadcasting* (FB) [6], *Skyscraper Broadcasting* [5], and *Mayan Temple Broadcasting* [17]. Some protocols use hybrid approach such that they partition each video into fixed size segments, and transmit them in fixed bandwidth channels by time division multiplexing. This category includes: *Pagoda Broadcasting* (PB) [14], *New Pagoda Broadcasting* [13]. However, these VoD protocols can only support normal playback of the video without offering VCR-like functions, for example, *forward jump* or *rewind*. They were optimized to take the advantage of regular video playback.

Several researches achieve interactive services for VoD transmitting MPEG coded videos. Chen *et al.* employ a P frame to I frame conversion [2] technique to support smooth reverse play. However, the conversion can only take place after the whole video has been downloaded. Lin *et al.* propose a dual-bit-stream method to support VCR functionality [9, 10]. This interesting technique requires two different copies for each video. *Backward* and *forward play*s are done through switching these two versions of video copies through the dedicated channel to the client. Vasudev designed in [19] a compressed domain MPEG transcoder that can be used to support reverse play. This technique also requires the server to store the normal and the reverse versions of the video.

Recently, Pâris proposed a technique called *Interactive Pagoda Broadcasting* (IPB) protocol [12] by extending their *Pagoda Broadcasting* (PB) protocol. This approach suggests to equip each interactive user with a disk buffer large enough to cache the entire video. Regular video playback is supported through receiving all the broadcast segments as usual. To enjoy interactive services, the users will preserve any video data received during the normal playback in the disk buffer. Individual VCR-like operations can therefore be easily fulfilled locally through the disk buffer. In case the segments have not been received during the regular playback, the media server will uses segment patching to supplement these missing segments in time. This interesting protocol is a feasible approach.

Extending from the PB, the drawback of IPB is high requirements in the communication and storage bandwidths of clients. In spite of huge aggregate download speed, there is still a large amount of video data unavailable during the regular playback. As a consequence, the number of video segments delivered by the server through the patching stream is considerable. The major cause results from the ill-managed broadcast scheduling such that even each client receives all the broadcast streams as quickly as possible, most of loading bandwidth is in fact useless. To improve the efficiency, we propose in this paper an alternative solution called *Cost-Effective Broadcasting* protocol (CEB) to reduce such high bandwidths demand. Our technique is built on the top of the *Striping Broadcasting* (SB) protocol [18], which employs a broadcast series {1, 2, 4, 8, …} as *Fast Broadcasting* (FB) [6] and *Client-Centric Approach* (CCA) [1, 4] do. Noticeably, SB elegantly defers the different broadcast streams with the proper phase offset to their broadcast periods. With this unique design, SB can require only *three* loaders for every client to support jitter-free playback. In this paper, we modify the striping protocol to support *forward jump* as IPB. As the performance study will show, despite much lower downloading rate, the proposed CEB technique can fully utilize such capacity to prefetch video segments beforehand. Compared to IPB, CEB may require less additional patching bandwidth, while using a significantly less disk storage bandwidth.

3. The Cost-Effective Broadcasting Protocol

In this section, we first discuss the design of the original SB technique. We then present the way to extend this technique to support the interactive services, such as *forward jump*. Subsequently, we will analyze the server patching bandwidth requirements, and mathematically compute the amount of video segments required for the techniques.

3.1 Original Striping Broadcasting Protocol

The *Striping Broadcasting* (SB) [18] divides the server bandwidth *B* into *M* sections of equal capacity, each dedicated for one video title, where *M* is the number of video titles to be broadcast. For each movie, the bandwidth is further partitioned into multiples of the video consumption rate, each denoted as a logical channel. In other words, the number of channels, *K*, for each video equals

$$K = \lfloor B / (M \cdot b) \rfloor \qquad (1)$$

where *b* is the video consumption rate.

Accordingly, every video of length *D* minutes is then segmented into *K* segments of increasing size with factor *two*. Therefore, the duration of the first segment is equal to

$$D_1 = \frac{D}{1 + 2 + \cdots + 2^{K-1}} = \frac{D}{2^K - 1}. \qquad (2)$$

The i^{th} video segment $(1 \le i \le K)$, denoted as S_i, is then periodically broadcast on the dedicated Channel *i*.

To avoid the worst case (all segments are aligned as in FB and CCA) that needs to listen to all the channels simultaneously[1] for jitter-free playback, the striping broadcasting technique introduces the offset idea to defer the broadcast schedules of the channels with the proper delays. As shown in Figure 1, S_i $(1 < i \le K)$ has an offset equal to half of its broadcast period. Traditional broadcasting protocols, like FB shown in Figure 1(a), occasionally start the segment broadcasts at the same time. As a result, clients need to tune up to all the channels for the desirable video segments; or, the jitter-free playback

[1] See Figure 1(a) for the example when the user starts to receive data in time slot 1

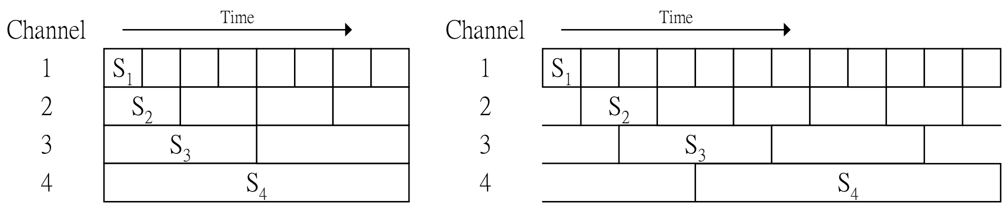


Figure 1. The illustration of the phase offset concept in SB with 4 channels

cannot be guaranteed. In contrast, SB staggers the broadcasts of video segments apart, as shown in Figure 1(b). As a result, no matter how many simultaneous broadcast streams (namely, *K*) are, every client needs only tune to at most *three* channels at the same time, as proved in [18]. Comparatively, most of the other broadcasting protocols, like FB [6] or PB [13, 14], need the concurrent reception of all *K* broadcast streams to guarantee jitter-free playback.

SB employs in every client *three* loaders to take turns in receiving segment broadcasts. To direct the access of all segments prior to their playbacks with only *three* loaders, SB utilizes one additional marker called "*tag*" (an integer between 0 and 2^{K-1}-1) to indicate the start slot of the last segment. An algorithm in [18] can derive quickly the tuning schedule from this tag for each client. Following the tuning schedule, the three loaders can tune to receive segment broadcasts in order. We note that each bit in the binary format of the tag represents one channel from 2 to *K*. For instance, if the tag were 6 (110 in binary), the turning order would be {1, 3, 2, 4}: S_1, S_3, S_2, & then S_4. Figure 2 illustrates such an example. A user arriving during the sixth time slot will start to receive segments with tag 6. Following the tuning order {1, 3, 2, 4}, the shaded segments will be received, where the numbers inside the shaded boxes indicate the receive order for this user.

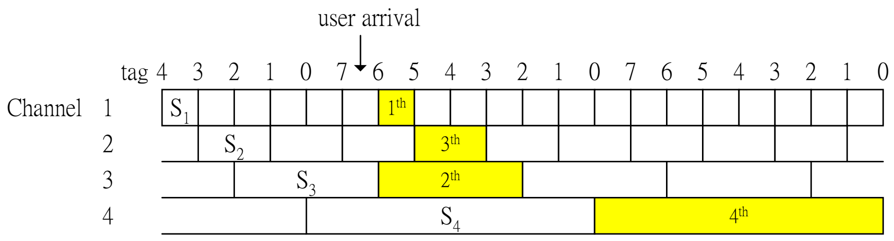


Figure 2. The tags of the SB protocol with 4 channels

To further minimize the client buffer space requirement, SB can optionally strip the last *K-N* segments (with size bounded at $2^N \times D_1$ minutes) into two sub-segments each. Then, it uses the channels of half consumption rate to broadcast them with an offset equal to half their duration, as shown in Figure 3. (Note that the tag is now between 0 and 2^N-1, indicating the start of the first striped segment). The disk space overhead in prefetching the last segment and thus the space requirement can be further reduced [18].

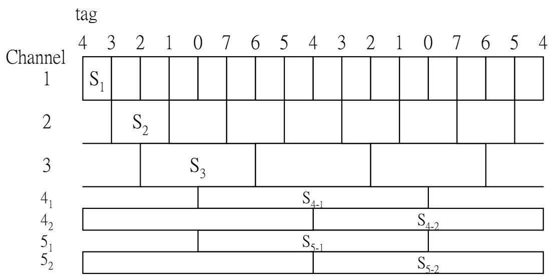


Figure 3. The striping protocol, N = 3, K = 5

3.2 The Cost-Effective Interactive Broadcasting Protocol

We employ the same offset idea of SB using only *three* loaders in our design to reduce the client communication bandwidth requirement from *K* times to thrice of the video consumption rate *b*, where *K* is the number of channels or simultaneous segment broadcasts. Since the goal of our new design is to support interactive video playback, client disk buffer needs to keep all the frames as long as needed. The striping mechanism for buffer size minimization is out of our concern. Therefore, we do not adopt the striping approach in the proposed technique. Rather, we retain the tag markers and its associated tuning algorithm to determine the jitter-free turning order. So we can still support normal playback.

We observe that the three loaders in the SB are not fully active. Sometimes, they may become idle awaiting the coming-in of the broadcast streams. That is because the loaders were designed to receive the segments starting from their beginning. To fully utilize such receiving capability to facilitate the *forward jump* function, we propose to receive any segment stream segments as early as possible. Specifically, we plan to utilize the idle time of these three loaders, which is the main idea of our new protocol.

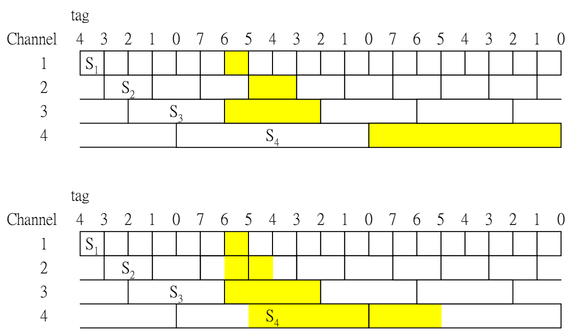


Figure 4. The Best Effort Receive Scheme, tag = 6, turning order: 1 3 2 4

Figure 4 illustrates the scenario using the previous example. Suppose one user arriving at time slot 6 will wait until the start of the first segment, and then start to receive data with tag 6. After the computation of tuning schedule, the turning order is determined as {1, 3, 2, 4}. In the original SB scheme, this user will receive segments S_1, S_3, S_2, & then S_4, marked as the shaded boxes in Figure 4(a). It is easy to see from the figure that Loader 3 for receiving S_2 will idle 1 time slot (time slot 7), while Loader 1 receiving S_4 will idle 5 time slots (time slot 8 to 12). Under the new approach, this user will receive segments S_1, S_3, S_2, & S_4 as soon as possible. Figure 4(b) shows the scenario of such earliest reception in terms of shaded boxes. Notice that these three loaders are now fully active in receiving video segments at their earliest possible. This important property will decrease the number of missing segments during segment patching if the user would like to perform *forward jump* later.

With the same turning order, each segment can be received on time or even earlier, our *Cost-Effective Broadcasting* (CEB) protocol can still ensure jitter-free playback. In addition to our best-effort receiving scheme, we use segment patching to assist our VCR support - *forward jump*. The missing segment data will be delivered from the server to the requesting client on demand.

We currently consider three types of VCR services: *pause*, *rewind* and *forward jump*. Specifically, the *rewind* signifies that users can jump back to a play point that they have already watched. Similarly, the *forward jump* means that users can jump forward to a future playback point and start normal playback therein.

Like the traditional VoD broadcasting techniques, we suppose the client *set-top box* (STB) has enough disk buffer space to store the whole video. Thus, users have fully interactive controls over the entire video. In addition, we also assume that the server reserves certain amount of server bandwidth to support user interactive (VCR) requests [8].

The supports for *rewind* & *pause* can be easily achieved from the video data stored earlier in the client local buffer. Similarly, *forward jump* can be offered from the disk buffer provided the data were saved during the video session. The server will supplement the missing data required for the *forward jump* operation by patching streams. If users want to pause, we let the loaders continue downloading segment data. Once users resume, users will consume the data already in disk.

3.3 Analytical Study

To simplify the patching duration computation, we assume that the VCR requests for each user can only take place once. This assumption is also adopted in [12].

Suppose the user have watched S_i, and decides to jump forward to watch S_j. In IPB, the probability that S_j will be received just in time or already stored in disk buffer is

$$\frac{i+1}{s(j)} \qquad (3)$$

where $s(j)$ is the broadcast period for S_j. The average patched segments sent for such user who jumps from S_i to S_j and then stays normal playback till the end is thus

$$\sum_{l=0}^{n-j}(1-\min(1,\frac{i+l+1}{s(j+l)}))\times 1. \qquad (4)$$

In the proposed CEB technique, with only three loaders, we cannot receive all channels at the same time. In case of jumping from S_i to S_j, the probability that S_j will be received just in time or already saved in disk buffer becomes

$$\frac{i+1-x(j)}{s(j)} \qquad (5)$$

where $x(j)$ is the delay slots of the reception process for S_j. We note that the period of S_j for CEB is

$$2^{\lceil log_2(j+1)\rceil - 1}. \qquad (6)$$

The average patched segments sent from the server to the interactive client is then

$$\sum_{l=0}^{n-j}(1-\min(1,\frac{i+l+1-x(j+l)}{2^{\lceil \log_2(j+l+1)\rceil - 1}}))\times 1. \qquad (7)$$

4. Performance Study

To assess the effectiveness of the proposed CEB technique, we perform intense investigation over a wide range of system settings. We wrote an emulation program to compute the correct value of $x(j)$ used in Formula (5) and (7) for each S_j given server bandwidth of K broadcast channels through searching all possible tag values. This program also computes the average patched segments using Formula (4) and (7).

Figure 5 compares the initial latency of *Fast Broadcasting* (FB) and *Interactive Pagoda Broadcasting* (IPB). The proposed technique CEB based on the SB technique will have the same performance as FB. Due to more precise segment-to-slot mapping, IPB can partition the video into more segments than FB given the same server channels, resulting in smaller initial latencies. However, the difference is unnoticeable when server bandwidth is large enough (for example, the difference is less then 14 seconds given 8 channels).

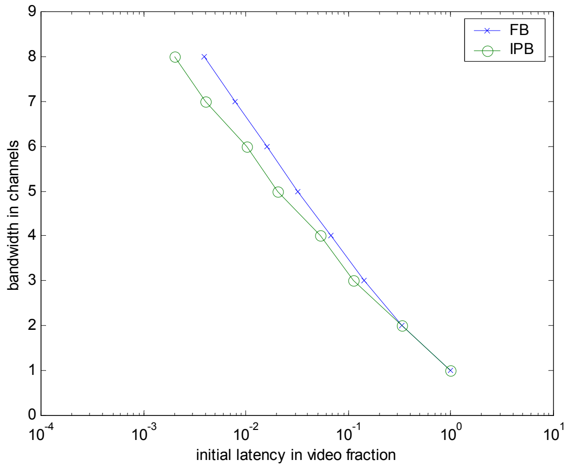


Fig. 5: Initial latency comparison of FB and IPB.

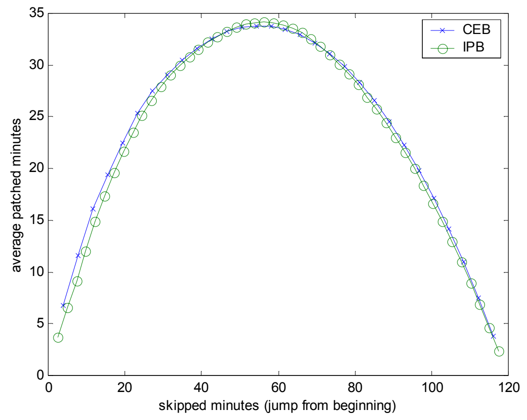


Fig. 6: Average patching (jump from beginning).

The comparisons between the average patching duration caused by *forward jump* is shown in Fig. 6. Since the segment sizes of CEB and IPB are not the same, we normalized them in unit of minutes to clarify the performance differences, where each video is assumed of 120 minutes long. The jumps start from the beginning of the video and to different target segments when the server bandwidth equals 5 times the video consumption rate. We can see from the figure that our scheme performs slightly better

Server bandwidth (channels)	3	4	5	10	15	20	30	40	50
IPB	18	22.5	27	49.5	72	94.5	139.5	184.5	229.5
CEB	18	18	18	18	18	18	18	18	18

Table 1. Client I/O bandwidth comparison (unit: Mbps)

when the jump distance is between 50 and 70 minutes and slightly worse otherwise. Generally speaking, both schemes perform comparatively. However, the proposed CEB technique employing only three loaders can achieve a better patching requirement, while the IPB demands to receive all channels, namely, five loaders.

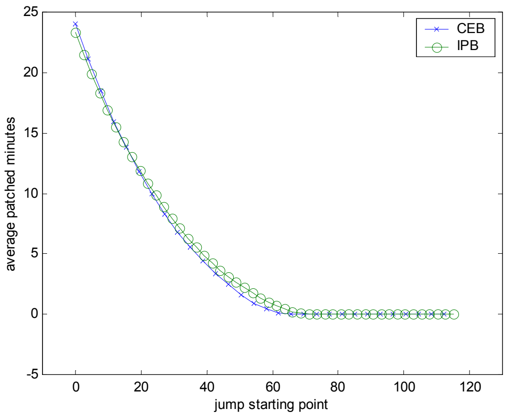


Fig. 7: Average patching (jump from all segments).

Figure 7 investigates for both CEB and IPB schemes the average patching duration from different jump starting points to all targets given the server bandwidth of five channels. We assume that the probability for each segment to become the jump target is equally likely. As shown in Figure 7, as the starting point of the jump progresses, the average patched data drops. The proposed CEB scheme slightly outperforms IPB. Notice that we only need 3 loaders while IPB needs five. The figure also indicates that the patched data finally drops to zero. This is because as soon as all segments have been received, no additional patching is required. We can receive all segments in 69 minutes 41 seconds by three loaders. (The last segment will be received 2 slots later in the worst case. In other words, 18 slots are needed.) In contrast, IPB needs the average of 73 minutes 29 seconds to finish the download of the entire video. (The last segment contains 30/49 portion of the video.)

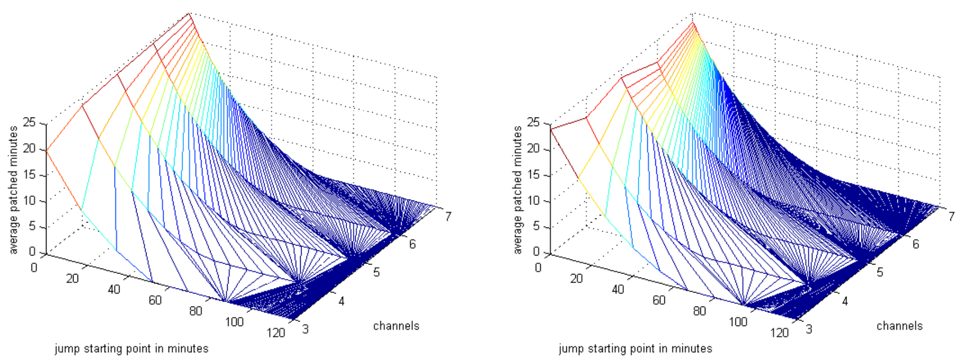
Figure 8 compares the average patching duration of CEB and IPB under various server bandwidth capacities. The bandwidth varies from 3 to 7 broadcast channels. As shown in the figure, the bandwidth did not significantly impact the patching duration for both schemes (the shape of the five curves are very similar). The patching duration drops gradually from about 25 minutes to zero. IPB performs slightly better at higher server bandwidth, but the curves of CEB drops faster then IPB since we can receive all segments earlier. (Our last segment is shorter than the one in pagoda broadcasting.) Notice that merely with three loaders, our scheme CEB can compete with IPB.

Table 1 lists the client disk storage bandwidth needed when the video is encoded using MPEG-2 videos with the playback rate of 4.5Mbps under different server bandwidth. Our scheme can achieve very low constant client storage I/O bandwidth, namely, 4 * 4.5 Mbps: three for loaders to receive data, one for the STB to render video. As the server bandwidth increases, IPB requires clients to have increasingly higher disk I/O access bandwidths.

5. Concluding Remarks

Conventional VoD broadcasting protocols were optimized to only support normal playback of videos. Their objectives are to minimize user initial latency, client STB buffer, etc. However, they did not offer interactive services, like *forward jump* or *rewind*.

We proposed in this paper a cost-effective broadcasting protocol derived from the unique features of the striping broadcasting technique. The proposed technique merely utilizes the *three* active loaders fully to accumulate video segments as early as possible. We then exploit these segments received ahead to help the *forward jump* operation. Compared to the recently proposed IPB, our scheme can offer the same level of interactive services, only requiring significantly less client communication and storage bandwidths.



(a) CEB (b) IPB

Figure 8. Average patched duration comparison of CEB and IPB under different server bandwidth.

References

[1] Y. Cai, K. A. Hua and S. Sheu, “Leverage Client Bandwidth to Improve Service Latency in a Periodic Broadcast Environment,” to appear in Journal of Applied Systems Studies.

[2] Ming-Syan Chen and Dilip D. Kandlur, “Stream Conversion to Support Interactive Video Playout,” IEEE Multimedia Magazine, 3(2):51-58, Summer 1996.

[3] Ailan Hu, “Video-on-Demand Broadcasting Protocols: A Comprehensive Study,” In Proc. Of IEEE Infocom 2001.

[4] K. A. Hua, Y. Cai, and S. Sheu, “Exploiting Client Bandwidth for more Efficient Video Broadcast,” In Proc. Of Computer Communications and Networks (IC3N’98), Lafayette, Louisiana, Oct. 1998.

[5] Kien A. Hua, Simen Sheu, “Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems,” In Proc. Of ACM SIGCOMM ’97 Conference, pages 89-100, Sept. 1997.

[6] L. Juhn and L. Tseng, “Fast data broadcasting and receiving scheme for popular video service,” IEEE Transactions on Broadcasting, 44(1):100--105, 1998.

[7] L. Juhn and L. Tseng, “Harmonic broadcasting for video-on-demand service,” IEEE Trans. on Broadcasting, 43(3):268--271, Sept. 1997.

[8] Wanjiun Liao and Victor O. K. Li, “The split and merge (SAM) protocol for interactive video-on-demand systems,” In Proc. of the 16th IEEE INFOCOM'97, Kobe, Japan, April 1997, pp.1349-1356.

[9] Chia-Wen Lin, Jian Zhou, Jeongnam Youn, and Ming-Ting Sun, “MPEG video streaming with VCR functionality,” IEEE Trans. Circuits and Systems for Video Technology, vol. 11, no. 3, pp. 415-425, Mar. 2001. (special issue on Internet streaming video)

[10] Chia-Wen Lin, Jeongnam Youn, Jian Zhou, Ming-Ting Sun and Iraj Sodagar, “MPEG video streaming with VCR functionality,” in Proc. of IEEE Int. Symp. Multimedia Software Eng., pp. 146-153, Dec. 2000, Taipei, Taiwan.

[11] A. Mahanti, D. L. Eager, M. K. Vernon, and D. S. Sundaram-Stukel, “Scalable On-Demand Media Streaming with Packet Loss Recovery,” In Proc. Of ACM SIGCOMM’01, pp. 97-108, San Diego, CA, August 27-31, 2001.

[12] Jehan-Francois Pâris, “An Interactive Broadcasting Protocol for Video-on-Demand,” In Proceedings of the 20th IEEE International Performance, Computing, and Communications Conference (IPCCC 2001), Phoenix. AZ, April 2001, pp.347-353.

[13] J. F. Pâris, “A Simple Low-Bandwidth Broadcasting Protocol for Video-on-Demand,” In Proc. of the first Conf. on Computer Communications and Networks, pp.690-697, Oct. 1999.

[14] Jehan-Francois Pâris, Steven W. Carter, and Darrell D. E. Long, “A hybrid broadcasting protocol for video on demand,” In Proc. of SPIE Multimedia Computing and Networking, pages 317-326, San Jose, California, January 1999.

[15] J.-F. Pâris, Steven W. Carter, and Darrell D. E. Long, “A low bandwidth broadcasting protocol for video on demand,” In Proc. of the Int’l Conf. on Computer Communications and Networks, October 1998.

[16] J.-F. Pâris, S. W. Carter, and D. D. E. Long, “Efficient broadcasting protocols for video on demand,” In Proc. of the 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98), pp. 127-132, July 1998.

[17] Jehan-Francois Pâris, Darrell D. E. Long, Patrick E. Mantey, “Zero-Delay Broadcasting Protocols for Video-on-Demand,” In Proc. of the Int’l Conf. ACM Multimedia, 1999, Orlando, FL, USA, pp. 189-197.

[18] S. Sheu, K.A. Hua and Y. Cai, “A dNel Broadcast Technique for Theaters in the air,” In Proc. of WVUME' 2000, pp. 218-225, Chicago, IL, July, 2000.

[19] B. Vasudev, “Compressed-domain reverse play of MPEG video streams,” In Proc. of SPIE Conf. Multimedia Systems and Applications, Nov. 1998, pp. 237-248.

[20] S. Viswanathan and T. Imielinski, “Pyramid Broadcasting for Video-on-Demand Service,” In Proc. of the SPIE Multimedia Computing and Networking, Vol. 2417, San Jose, CA, February 1995, pp. 66-77.