Providing VCR Functionality in Staggered Video Broadcasting[⋆]

Jin B. Kwon and Heon Y. Yeom

School of Computer Science and Engineering
Seoul National University
Seoul, South Korea 151-742
{jbkwon,yeom}@dcslab.snu.ac.kr

Abstract. A true video-on-demand(TVOD) system lets users view any video program, at any time, and perform any VCR functions, but its per-user video delivery cost is too expensive. A near video-on-demand(NVOD) is a more scalable approach by batching multiple clients to a shared stream or broadcasting videos. Staggered video broadcasting, one of NVOD techniques, broadcasts multiple streams of the same video at staggered times, with one stream serving multiple clients. In order to provide subscribers with a high-quality VOD service, it is desirable to add VCR functionality such as fast forward and fast backward, but it is not easy to provide VCR functionality in NVOD, especially video broadcasting system where any dedicated or interaction channel is not available.
In this paper, we analyze the conditions necessary to provide VCR functions and then propose a reception schedule which satisfies these conditions, with minimal resource requirements. Since our proposed scheme receives video frames as a unit it can keep up rapidly with a changing VCR action pattern. It is demonstrated that the scheme provide VCR functionality consistently through simulations.

1 Introduction

Video-On-Demand (VOD) service enables a subscriber to watch a video of his choice whenever he wants. In True-VOD (TVOD) systems, each subscriber is served by an individually allocated channel. Although TVOD can respond to requests immediately, server network bandwidth runs out rapidly. Thus, unicast VOD systems are expensive due to their low scalability. It is known that the majority of the requests are for a small group of videos and that the popularities of these videos follow the Zipf distribution[9, 10]. The network I/O bottleneck presented by TVOD may be eliminated by employing a multicast facility of modern communication networks[1, 8, 16, 20] to share a server stream among multiple clients. These services are referred to as Near-VOD (NVOD) service.

There are two basic approaches to NVOD provision. One is called *scheduled multicast* and the other *periodic broadcast*. In conventional scheduled multicast[4,

[⋆] This work wass supported by the Brain Korea 21 Project in 2001

5, 7, 9, 10, 21], the server collects user requests (i.e., a *batch*) during a specific time period. Clients requesting the same video within the same period will receive the video stream over a single multicast channel. When a server channel becomes available, the server selects a batch to multicast according to some scheduling policy. For instance, the *Maximum Queue Length* (MQL)[9] selects the batch with the highest number of pending requests to serve first. Periodic Broadcasts, as the name suggests, broadcasts videos periodically and can service an unlimited number of clients simultaneously with bounded service latency [9]. That is, a new stream corresponding to a video starts every d seconds, and the server channels are individually allocated to each video. Periodic broadcast schemes guarantee the maximum *service latency* experienced by any client to be less than d seconds, and allocate one or more channels to each video. Periodic broadcast bounds the service latency, bypasses the need to process individual user requests, and eliminate the need for an upstream channel. Thus, periodic broadcast is more scalable than TVOD or other NVOD techniques. Due to these benefits, a number of periodic broadcast schemes have been recently presented[3, 9, 12–14, 17–19, 22]. In *staggered broadcasting*[9], several channels broadcast a video periodically with staggered start times. In this case the maximum service latency is the length of video divided by the number of the channels allocated for the video. Most proposed schemes aim at minimizing the system required resources for a given maximum service latency, or minimizing the maximum service latency for a given system resource, such as server network bandwidth, client I/O bandwidth, client disk space, etc. Recently, videos have been fragmented into separate segments and each segment then transmitted repeatedly over a different channel. Periodic broadcast schemes can be divided into *pyramid-based* schemes and *harmonic-based* schemes. The pyramid-based schemes[3, 12, 13, 22] divide each video into segments of "increasing size" and transmit the segments over "equal bandwidth" channels. On the other hand, the harmonic-based schemes[14, 17, 18] divide each video into segments of "equal size" and transmit the segments over "decreasing bandwidth" channels.

Digital television(DTV) technology appears commercially today in digital video broadcasting systems, such as digital satellite, Cable TV(CATV), and terrestrial broadcasting. The key benefit of DTV is the high transport efficiency - digital compression packs five or more times as many channels in a given distribution-network bandwidth. This makes it possible to delivery more content and pay-per-view events with multiple closely spaced start time(i.e., staggered broadcasting). This trend makes periodic broadcast more feasible than scheduled multicast. In order to provide subscribers with a high-quality video service, it is desirable to add VCR functions such as fast forward or fast backward to NVOD services. Since TVOD allocates a channel to each client, it is relatively easy to provide such VCR functions. However, in NVOD it is not easy to provide VCR functions due to the characteristics of the multicast. Several schemes have been proposed to deal with the problem of providing VCR functions in NVOD systems[2, 5–7, 15]. However, most of these have addressed VCR functions in scheduled multicast systems and depend on both a client buffer and "interactive" channels to provide VCR functions. Client buffering techniques can be applied to the periodic broadcast model, but interactive channels are not available. Fei et al. proposed a scheme to provide VCR functions in a staggered broadcast[11]. However, the scheme cannot guarantee VCR functions. In other words, it cannot determine whether a requested VCR action can be provided. Moreover, the scheme cannot rapidly adapt to the user's pattern of VCR actions because of its *segment-level* reception schedule.

In this paper, we present the conditions required for providing consistent VCR actions in NVOD systems using staggered broadcasting, and then propose a reception schedule for staggered broadcasting which requires minimal resources while satisfying the conditions.

The remainder of this paper is organized as follows. First, we introduce some previously published related work in Sect. 2. And, in Sect. 3 we present the conditions for continuous VCR functions theoretically, and propose a reception schedule for staggered PB, while satisfying these conditions. Section 4 demonstrates the effectiveness of the proposed schemes through simulations, and Sect. 5 summarizes the performance of our scheme and includes a discussion on some other issues. The paper concludes with Sect. 6.

2 Related Work

NVOD satisfies the requests of several clients with one channel and thus circumvents the the need for individual subscriber service. Although it is desirable to provide VCR functionality for high-quality service, it is difficult in NVOD systems since clients do not have dedicated channels. Several schemes have been proposed to deal with the problem of providing VCR functions in NVOD systems[2, 5–7, 15]. Almeroth and Ammar incorporated the VCR functions into NVOD systems and introduced the concept of discontinuous VCR actions. The scheme uses client buffering to provide VCR functions and proposes that *emergency interactive channels* be used when the client buffer contents are insufficient for the desired interaction. The SAM (*Split And Merging*) protocol[15] uses a synch-buffer and special interactive channels, called I-stream, to provide VCR functions. Abram-Profeta and Shin improved the SAM protocol by changing the shared synch-buffers to separate buffers at each client, thus making the system more scalable[2].

Most of these proposed schemes have addressed VCR functions in scheduled multicast models and depend on both the client buffer and interactive channels to provide VCR functions. Client buffering techniques can be applied to the periodic broadcast model, but interactive channels are not available. Even if interactive channels are available, it is not preferable because using interactive channels compromises the scalability of the NVOD service. As an alternative to interactive channels in the periodic broadcast model, VCR functions that cannot be covered by the client buffer can be accommodated by switching channels and receiving from multiple channels. Fei et al. proposed Active Buffer Management (ABM), a client buffer management scheme, to provide VCR functions in staggered PB[11]. This scheme focuses on raising the probability of servicing users' VCR requests with a fixed client buffer. However, when using existing client buffering schemes[2, 5–7, 15], consecutive VCR actions in the same direction(forward or backward) result in service disruption since the play point will ultimately move to the boundary of the buffer. At this time and in this direction, the VCR functions can no longer be provided. The idea of ABM is to keep the play point in the middle of the buffer so that there is a lower probability that VCR actions will move the play point beyond the buffer capability. In ABM, a buffer manager adjusts the contents of the buffer after VCR actions so that the relative position of the play point in the buffer remains central. The buffer manager is presumed to be able to receive data from three channels simultaneously. When the broadcast of each segment is finished, i.e.,$t_0 + k \cdot d$, the buffer manager decides which segments it will receive during the next d seconds, based on the position of the play point. However, although ABM increases the likelihood that VCR actions may be provided, it did not address the method of determining whether a request can be provided. Moreover, it receives the whole segment or nothing and the decision on reception schedule is invoked every d seconds. Thus, since it cannot adapt itself rapidly to a change in the user's VCR action pattern, the probability of VCR implementation may become low.

In this paper, we analyze the conditions necessary to guarantee VCR functions and then propose a reception schedule which satisfies these conditions, with minimal resource requirements. Since our proposed scheme receives video frames as a unit, unlike ABM, it can keep up rapidly with a changing VCR action pattern. The symbols that are used in the next stage of th presentation are shown in Table 1.

Table 1. Notations

S_i	ith segment
d	length of segment (sec.)
s	size of segment (KB)
K	number of segments
γ	play rate(fps)
b	channel bandwidth required for γ

3 VCR Functions in Periodic Broadcast

3.1 Staggered Broadcasting

Staggered broadcasting starts video broadcasts over the allocated channels at fixed time intervals. Assume that there are K channels dedicated to a video of length L. The K channels are labeled as $0, 1, 2, \ldots, K - 1$. The video broadcast starts every $L/K(= d)$ seconds and each channel broadcasts the whole video

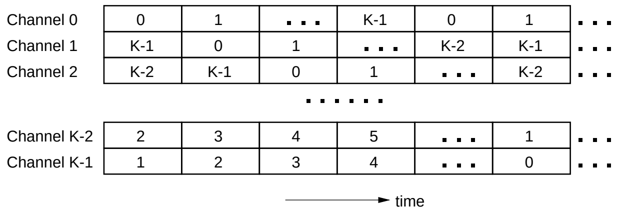

Fig. 1. Staggered Broadcasting

repeatedly as shown in Figure 1. If the broadcast of a video starts at channel 0 at the system setup time t_0, the broadcast starts over channel k at $t_0 + k \cdot d$ and is repeated. Therefore, the video is broadcast every d seconds over K channels and accordingly, the maximum service latency is d seconds. Here, the K video chunks of length d that make up the video are called *segments*. During a segment time(i.e., d seconds), all K segments are broadcast over different channels. And, we assume that broadcast of all the segments are synchronized at the level of frames. In other words, if the jth frame of a segment is broadcast at time t, the jth frames of the other segments are also broadcast at t.

3.2 VCR Functions

VCR functions include *play forward, play backward, fast forward, fast backward, slow forward, slow backward, jump forward, jump backward, pause*, and so on. We consider the following VCR functions in this paper:

1. Play Forward/Backward (PF/PB): Playback the video at a normal playback rate in either the forward or backward direction.
2. Fast Forward/Backward (FF/FB): Playback the video at n times the normal playback rate in either the forward or backward direction. We assume n is 3 for simplicity in this paper, but our work can be applied generally to FF/FB functions.
3. Slow Forward/Backward (SF/SB): Playback the video at a $1/m$ times the normal playback rate in either the forward or backward direction.
4. Pause (PA): Stop the playback for a period of time.
5. Jump Forward/Backward (JF/JB): Jump immediately to the destination frame.

The functions can be conveniently classified into *forward functions* and *backward functions*, and although Pause(PA) has no direction, it can be regarded as one. In addition, the functions were categorized as *continuous* and *discontinuous functions* according to continuity in a series of frames displayed as the result of each one, which does not involve temporal notion. JF and JB functions are discontinuous functions and the remainder are continuous. Although PA holds a frame on the screen for the duration paused, when subsequent playback is resumed, the next frames are displayed in regular order. Since any next frames are not skipped at resume. That is why we regard PA as a continuous function.

3.3 Conditions for Continuous VCR

There are physically only broadcasting streams or channels in PB model, but we can give users an illusion that they are being serviced by dedicated streams. These *virtual* streams of the users' view are possible by effective client-side buffering and prefetching, and they can provide VCR functions just as the dedicated streams of TVOD. We first derive the conditions for providing continuous VCR functions consistently and find theoretically the minimum buffer requirement satisfying the conditions. In Sect. 3.4, using these conditions, we propose a reception schedule that provides VCR functions with the minimum buffer requirement. Discontinuous functions are not considered in this subsection.

The video frame of a video currently accessed by a clients is known as the *play point*, and the frames already displayed are called "past" frames and those that have not been displayed yet are called "future" frames. The *forward* and *backward buffers* are defined as the buffers keeping the future and the past frames, respectively. Figure 2 illustrates the relation of video object, broadcast channel, and client buffer. $\Delta(k_0, k)$ is the *distance* between the play point k_o and a future

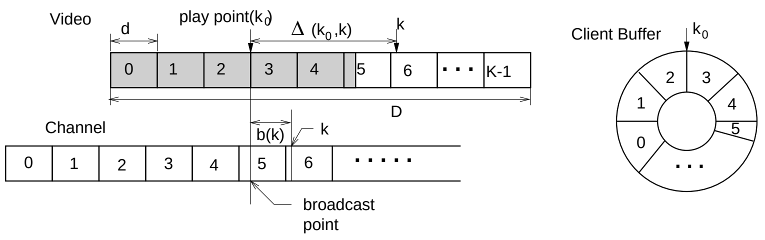

Fig. 2. Video, Segment, and Buffer

frame k at a normal play rate γ. For example, the distance between the first frame of S_0 and the first frame of S_2 is $2d$. Thus, the consumption time, $c(k)$, is the time remaining until frame k is consumed, and meets the condition

$$c(k) \ \geq \ \frac{\Delta(k_0, k)}{3}.$$

That is, $c(k)$ is greater than or equal to the time taken to consume all the frames between the frames k_o and k by FF or FB, whose play rates are three times the normal rate(i.e., 3γ). We thereby derive the condition for guaranteeing continuous functions as follows.

Theorem 1. *Let $\mathcal{B}$ be a set of frames contained by the buffer. Continuous VCR functions can be provided if the following condition is satisfied:*

$$\forall k \notin \mathcal{B}, \quad \frac{\Delta(k_0, k)}{3} \ \geq \ b(k), \tag{1}$$

where $b(k)$ is the next broadcasting time of a frame k.

PROOF If, for all k, we can satisfy

$$b(k) \leq c(k) \quad \text{or} \quad k \in \mathcal{B},$$

it is evident that continuous VCR actions can be provided. Since $c(k) \geq \Delta(k_0, k)/3$, $\Delta(k_0, k)/3 \geq b(k)$. Finally, Eq. 1 is valid. □

$\mathcal{B}_f$ is the set of frames contained by the forward buffer and $\mathcal{B}_b$ the set of those contained by the backward buffer. Since $\mathcal{B} = \mathcal{B}_f \cup \mathcal{B}_b$, for all the future frames and all the past frames respectively, Eq. 1 must be satisfied to provide continuous VCR functions. That is, for all future frames not in the forward buffer, their distances must be greater than or equal to $3b(k)$, and the situation for past frames is identical. The minimum buffer requirement for guaranteeing the continuous functions can be determined using THEOREM 1.

Lemma 1. *The minimal client buffer space required for a client to provide continuous VCR functions consistently is* $6s$*, where* s *is a segment size.*

PROOF In order to satisfy Eq. 1, all the frames k such that $\Delta(k_0, k) < 3$ must be in the buffer or be broadcast before $c(k)$. In the worst case, $3d\gamma$ future frames and $3d\gamma$ past frames must be in the buffer. Therefore, since data size of $d\gamma$ frames is s, buffer space of at least $6s$ is required. □

3.4 Reception Schedule

It is impossible to meet Eq. 1 for d seconds after a client begins to receive the video data. Since each segment is broadcast at a bandwidth of b, d seconds are required to receive the whole segment. Hence, the FF actions of S_0 and S_1 may not be serviced for d seconds since the service start. In order to provide the FF function fully, the reception of S_0 must be finished within $d/3$ seconds after the service start and that of S_1 should be finished within $2d/3$ seconds. As this reception is impossible in staggered broadcasting, we propose a reception schedule scheme to guarantee all the continuous functions *after* d *seconds*, with a reception bandwidth requirement of $3b$.

Our presentation in this subsection focuses on the continuous forward(CF) function, since almost identical considerations apply to the case of the continuous backward(CB) function. Since all frames are transmitted every d seconds, the maximum of $b(k)$ in Eq. 1 is d. Thus, all future frames k such that $\Delta(k_0, k) \geq 3$ do not have to be contained by the forward buffer. Therefore, when play point k_0 is a frame of a segment S_i, the target segments we have to be concerned with for the future frames are S_i, S_{i+1}, S_{i+2}, and S_{i+3}, and similarly the target segments for the past frames are S_i, S_{i-1}, S_{i-2}, and S_{i-3}. K segments are transmitted synchronously over K channels(Fig. 1). Accordingly, the jth frames of all K segment are transmitted at the same time. Fig. 3 illustrates the target channels transmitting the target segments and the candidate frames for reception. In the figure, the play point k_0 is a frame of S_5, and the shaded area means the frames whose distances from k_0 are less than $3d$. Thus, the target segments are S_5, S_6,

S_7, and S_8, and the candidate frames are k_1, k_2, and k_3. Our reception schedule scheme puts into the buffer the frames not yet buffered among the candidate frames. When the client is running CF actions, the candidate frames can be in

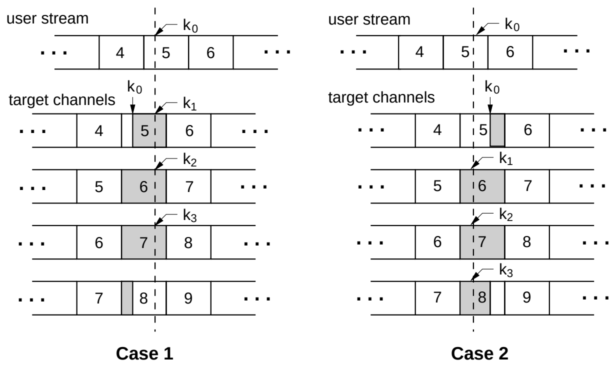

Fig. 3. Candidate Future Frames

two cases. Case 1 occurs when the frame index within segment of the play point is larger than those of the frames being currently broadcast, and Case 2 occurs in the other case(Fig. 3). Hence, the schedule receives data from at most three channels in both cases.

Since the client buffer needs to keep the frames with distances less than $3d$, the buffer requirement is $6s$(Lemma 1). The frames k such that $\Delta(k_0, k) \leq 3d$ are called *essential frames*. If Eq. 1 meets, all essential frames k such that $b(k) > c(k)$ should be in the buffer. Hence, the number of the essential future frames in the buffer is less than or equal to $3d\gamma$. That is, data size of these future frames is less than or equal to $3s$. Therefore, if the size of the forward buffer,$|\mathcal{B}_f|$, is larger than or equal to $3s$, all essential future frames k are kept in the buffer. The case of the backward buffer $\mathcal{B}_b$ is identical. Since $|\mathcal{B}_b| = 6s - |\mathcal{B}_f|$, if $|\mathcal{B}_f| < 3s$, the candidate past frames do not have to be received because they all would be in the buffer, and if $\mathcal{B}_f \geq 3s$, the future candidate frames do not have to be received. Finally, the required reception bandwidth of our scheme is $3b$. Fig. 4 summarizes our reception algorithm. Once a new frame is put into buffer according to the algorithm, the frame with the longest distance from the play point is replaced. Therefore, since the client buffer is required to keep only essential frames with the essential frames, the buffer requirement is $6s$ from Lemma 1.

Example Fig 5 illustrates how our scheme works with a scenario on user behavior. The figure shows reception schedule and change of the forward buffer size on the user behavior. When the service starts at t_0, the client begin to receive S_0, S_1,and S_2 and display S_0 simultaneously, and data is accumulated in the buffer at a rate of $\frac{2}{3}s$ per d seconds. At t_1, the forward buffer has S_1 and S_2. S_0 has been consumed and this is in the backward buffer. From t_1 to t_2 , the frames of S_3 and S_4 are the candidates, since if S_1 and S_2 are displayed by

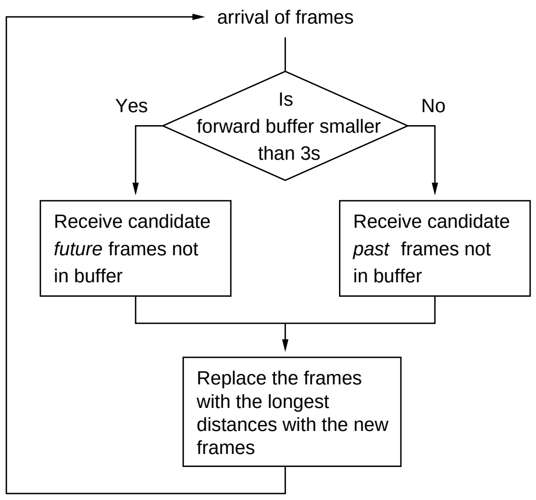
arrival of frames
Is forward buffer smaller than 3s
Yes
No
Receive candidate *future* frames not in buffer
Receive candidate *past* frames not in buffer
Replace the frames with the longest distances with the new frames

Fig. 4. Reception Schedule

a FF action, the frames of S_3 and S_4 are required before the next broadcast of them (i.e.,t_2). Hence, the reception scheme receives them from t_1 to t_2. At t_2, the buffer has three segments, S_2, S_3, and S_4, and the user performs a long FF action S_2, S_3, and S_4 to t_9. The client begins to receive S_5, S_6, and S_7 at t_2, t_3, and t_4, respectively. Then, S_5 are displayed by PF from t_5 to t_6, and simultaneously the client receives S_8. The forward buffer size does not exceed $3s$. □

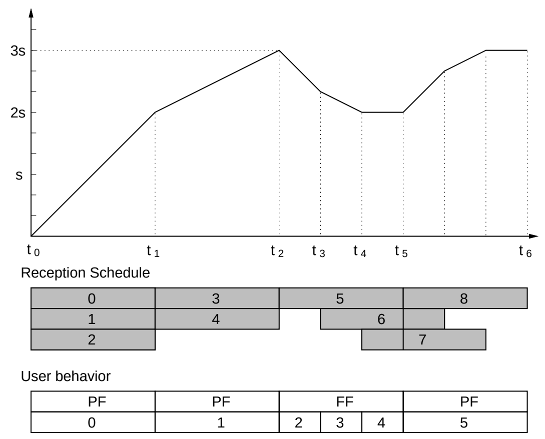

Fig. 5. Example

3.5 Discontinuous VCR Actions

Now we concern ourselves with the discontinuous functions such as JF and JB. Fig. 4 is the algorithm guaranteeing continuous functions without considering discontinuous functions. Discontinuous actions render the buffered content received according to the algorithm useless. Moreover, it is impossible to jump immediately to the requested destination due to buffer restrictions in client-side and service latency in the periodic broadcast. Nevertheless, it is necessary to provide continuous functions at some level of guarantee even after jump actions are performed. The client must be able to display the video by at least PF immediately after the jump. We aim at the same VCR provision as guaranteed when the service starts, except for continuous backward(CB) actions. CB actions do not have a meaning at service start, but they do in the case of jump actions. Thus, it need to be considered how to provide CB actions after a discontinuous action (e.g., their provision can be guaranteed after d seconds or after the nearest $d\gamma$ past frames are buffered). Since, however, the theoretical analysis is complicated and then more work is needed, we leave it to our future work. Instead, CB functions after jumps are provided by best effort according our scheme but not guaranteed in this paper.

As mentioned in Sect. 3.3, FF actions may not be serviced since Eq. 1 cannot be satisfied from service start until d seconds elapse. In order to provide PF functions consistently, the following condition must meet for all frame k such that $\Delta(k_0, k) \leq d$:

$$\Delta(k_0, k) \geq b(k) \quad \text{or} \quad k \in \mathcal{B}. \tag{2}$$

This condition meets at the service start ($k_0 = 0$) in staggered broadcasting. Therefore, the condition also must meet after jump action. In addition, since all the frames of a video are transmitted every d seconds, the same VCR provision as guaranteed at the service start can be possible.

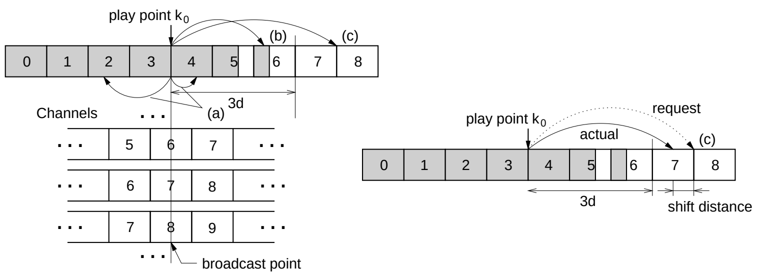

Fig. 6. Jump Actions and Destination Shift

Fig 6 illustrates how jump actions work for a snapshot of buffer state and play point. The shaded area means the frames kept in buffer and the snapshot is the state at t_4 in Fig. 5. The jump actions can be classified into three cases. (a) and (b) are the cases in which the destination points k_s are buffered, and (c) is the case in which the destination point is a non-essential frame not buffered. In case (a), since all frames k such that $\Delta(k_s, k) \leq d$ are kept in the buffer, (2) is satisfied and accordingly the jump to k_s is possible. On the other hand, in case (b) k_s is buffered but the condition is not satisfied, and in case (c) even k_s is not in the buffer. Thus, it is inevitable to jump to the frame nearest the requested destination point among the currently broadcasting frames in case (b) and (c). These are called *destination shift*. In summary, our scheme moves the play point to the requested frame k_s if, (2) is satisfied for future frames with distances less than d, and otherwise it moves the play point to the nearest frame k_t to k_s among frames being currently broadcasted. All the frames k such as $\Delta(k_s, k) \leq d$ need not to be checked for the condition. If a frame k_t within the range is broadcast before $\Delta(k_s, k_t)$ seconds and all the frames k such that $k_s < k < k_t$ satisfy (2), the other frames within the range also satisfy it. Let us jth frames of each segment are being broadcast over K channels, when the user requests the destination frame k_s. Then, k_s would lie between jth frames of any two successive segments. Since a segment length is d seconds, the distance between the two frames is d seconds. Hence, the maximum shift length between the requested destination and the actual one is $d/2$ and the average is $d/4$.

4 Simulation

In this section, we demonstrate the viability of our reception schemes for VCR functions through simulations. The simulations are made for two scenarios on user behavior, at which the user does only continuous actions (OCA) and all actions (ALL), respectively. Table 2 shows the probabilities that the user requests each action under OCA and ALL. Since there is no data recognized on users' VCR action pattern, we chose the probabilities arbitrarily. However, users' action pattern dose not affect the feasibility of our scheme. The holding duration of a

Table 2. User Action Patterns

	CONT.							DISC.	
	PF	PB	FF	FB	SF	SB	PA	JF	JB
OCA	1.00							0.00	
	0.32	0.06	0.32	0.10	0.08	0.04	0.08	0.00	0.00
ALL	0.80							0.20	
	0.26	0.05	0.26	0.08	0.06	0.03	0.06	0.10	0.10

continuous action is exponentially distributed with a mean of 10 seconds, and the jump distance of a discontinuous action are also exponentially distributed with a mean of 60 seconds for the ALL case. The jump distance is the distance between the current play point and the destination point. The segment length d is 60 seconds, the number of channels K is 32, and play rate γ is 30 fps. However,

K and γ do not make any effect on the simulation results, and are related only to the running time of the simulation.

Fig. 7 and Figure 8 show the variation of amount of buffered essential frames for OCA and ALL cases. We have not encountered any jitter under all the scenarios during simulation, since our schemes restrict some actions during the period when the actions are not guaranteed to be provided (e.g., FF actions are rejected for d seconds after the service starts). The results confirm that our reception schemes provide VCR functionality at the guarantee level which we presented in the previous sections. In addition, the results demonstrate require that the buffer requirement of the schemes is $6s$, which is the minimal buffer space required for providing VCR functions consistently. Figure 8 is the result for ALL case having 60 seconds as a mean jump distance. The larger the mean jump distance is, the more content buffered are corrupted by the jump and this is confirmed by the result. The vertical droppings of the line in the figure mean the corruption of buffered content.

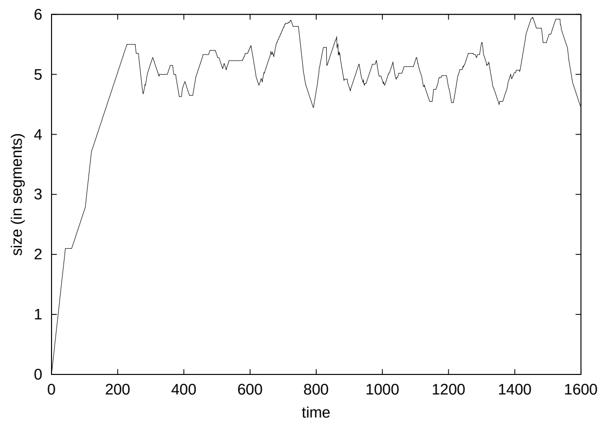

Fig. 7. Only Continuous Actions (OCA)

When a discontinuous action is requested, a destination shift may occur. Since the average shift distance is a important performance metric for the discontinuous actions, we observed the shift distances caused by jump actions during the simulations. Table 3 shows distance per shift, proportion of the actions causing no shift, and average shift distance, while varying mean distance between play point and the requested destination. The distance per shift is the average shift distance experienced by the actions having caused a shift, and the average shift distance is that of all jump actions. The distance per shift are about 15 seconds, respectively. Since d is 60 seconds, the result agrees with the analytic result that the average distance is $d/4$. As the mean jump distance is longer, the proportion of the actions without shift decreases and the average shift distance increases.

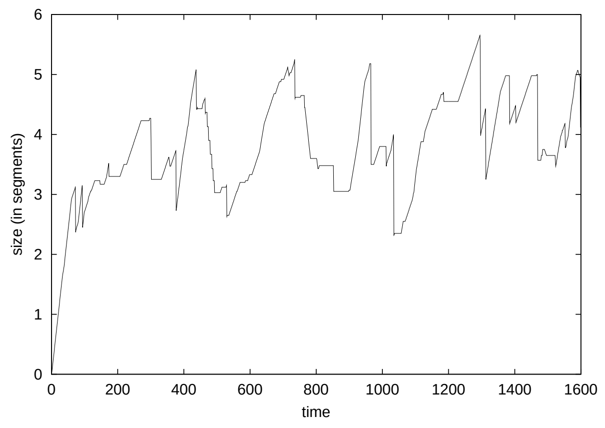

Fig. 8. ALL Actions (ALL, mean 60 sec.)

That is because long jump actions are not probably serviced by the buffered data.

Table 3. Destination Shift

mean dist	per-shift dist.	no shift	avg. shift dist.
30	13.76	0.83	2.34
60	16.94	0.58	7.11
90	15.32	0.35	9.96
120	17.11	0.21	13.52
150	15.92	0.10	14.33
180	15.82	0.09	15.82

5 Discussion

Since the proposed scheme works at the frame-level, the computational overhead caused by the algorithms must be reasonable. Otherwise, hiccup or jitter will occur on the screen by the overhead. Our reception schedule algorithm(Figure 4) have little computational overhead, which is $O(1)$. The algorithm for discontinuous actions require $O(d \cdot \gamma)$ time complexity, which is not great, and this can be reduced by optimization. Even if the overhead is a burden on the client, a little delay is acceptable due to the characteristics of discontinuous functions.

We have assumed that each channel is synchronized at the frame level. However, our schemes can be applied to the case that the channel is less frequently synchronized (e.g., every γ frames) by buffering a chunk of frames as a unit. Also, although we assumed three-times FF and FB functions, our schemes are easily extended to n-times FF and FB functions by replacing 3 with n in the equations presented throughout this paper.

6 Conclusion

A near video-on-demand(NVOD) is a more scalable approach by batching multiple clients to a shared stream or broadcasting videos. The advent of digital video broadcasting systems, such as digital satellite, CATV, etc, makes periodic broadcasting more feasible. Staggered video broadcasting, one of periodic broadcasting technique, broadcasts multiple streams of the same video at staggered times, with one stream serving multiple clients. In order to provide subscribers with a high-quality VOD service, it is desirable to add VCR functionality such as fast forward and fast backward, but it is not easy to provide VCR functionality in NVOD, especially video broadcasting service where any dedicated or interaction channel is not available.

In this paper, we analyze the conditions necessary to provide VCR functions and then propose a reception schedule which satisfies these conditions, with minimal resource requirements. Since our proposed scheme receives video frames as a unit it can keep up rapidly with a changing VCR action pattern. Our scheme makes it possible for users to enjoy the freedom of VCR actions without increasing the overall network bandwidth requirement, and requests only a little more buffer space and three times the bandwidth from the clients' side.

References

1. IEEE Standard 802.6. Distributed Queue Dual Bus (DQDB) Metropolitan Area Network (MAN), December 1990.
2. Emmanuel L. Abram-Profeta and Kang G. Shin. Providing Unrestricted VCR Functions in Multicast Video-On-Demand Servers. In *Proc. of IEEE International Conference on Multimedia Computing and Systems*, pages 66–75, Austin, Texas, June 1998.
3. C.C. Aggarwal, J.L. Wolf, and P.S. Yu. A Permutation-based Pyramid Broadcasting Scheme for Video-on-Demand Systems. In *IEEE International Conference on Multimedia Computing and Systems(ICMCS'96)*, pages 118–126, Hiroshima, Japan, June 1996.
4. C.C. Aggarwal, J.L. Wolf, and P.S. Yu. On Optimal Batching Policies for Video-On-Demand Storage Servers. In *IEEE International Conference on Multimedia Computing and Systems(ICMCS'96)*, Hiroshima, Japan, June 1996.
5. Kevin C. Almeroth and Mostafa Ammar. A Scalable Interactive Video-On-Demand Service Using Multicast Communication. In *Proc. of International Conference of Computer Communication and Networks (ICCCN'94)*, San Francisco, California, September 1994.
6. Kevin C. Almeroth and Mostafa Ammar. On the Performance of a Multicast Delivery Video-On-Demand Service with Discontinuous VCR Actions. In *Proc. of*

International Conference on Communication (ICC'95, Seattle, Washington, June 1995.

7. Kevin C. Almeroth and Mostafa Ammar. On the Use of Multicast Delivery to Provide a and Interactive Video-On-Demand Service. *IEEE Journal of Selected Areas in Communications*, 14(6):1110–1122, 1996.
8. J. Y. L. Boudec. The Asynchronous Transfer Mode: A Tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.
9. A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-demand Video Server with Batching. In *Proc. of ACM Multimedia*, pages 15–23, Oct 1994.
10. A. Dan, D. Sitaram, and P. Shahabuddin. Dynamic Batching Policies for an On-demand Video Server. *Multimedia Systems*, 4(3):112–121, June 1996.
11. Zongming Fei, Ibrahim Kamel, Sarit Mukherjee, and Mostafa H. Ammar. Providing Interactive Functions for Staggered Multicast Near Video-On-Demand Systems (Extended Abstract). In *Proc. of IEEE International Conference on Multimedia Computing and Systems(Poster Session)*, volume 2, pages 949–953, Florence, Italy, June 1999.
12. Lixin Gao, Jim Kurose, and Don Towsley. Efficient Schemes for Broadcasting Popular Videos. In *Proceedings of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '98)*, Cambridge, UK, July 1998.
13. K.A. Hua and S. Sheu. Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems. In *ACM SIGCOMM '97*, pages 89–100, Cannes, France, September 1997.
14. L. Juhn and L. Tseng. Harmonic Broadcasting for Video-on-Demand Service. *IEEE Transactions on Broadcasting*, 43(3):268–271, September 1997.
15. Wanjiun Liao and Victor O. Li. The Split and Merge Protocol for Interactive Video-On-Demand. *IEEE Multimedia*, 4(6):51–62, 1997.
16. D. J. Marchok, C. Rohrs, and M. R. Schafer. Multicasting in a Growable Packet (ATM) Switch. In *Proc. of IEEE INFOCOM*, pages 850–858, Bal Harbour, Florida, 1991.
17. J.-F. Pâris, S.W. Carter, and D.D.E Long. A Low Bandwidth Broadcasting Protocol for Video on Demand. In *IEEE International Conference on Computer Communications and Networks (ICCCN'98)*, pages 690–697, October 1998.
18. J.-F. Pâris, S.W. Carter, and D.D.E Long. Efficient Broadcasting Protocols for Video on Demand. In *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'98)*, pages 127–132, July 1998.
19. J.-F. Pâris, S.W. Carter, and D.D.E Long. A Hybrid Broadcasting Protocol for Video on Demand. In *Proc. of Multimedia Computing and Networking Conference (MMCN'99)*, pages 317–326, January 1999.
20. M. A. Rodrigues. Erasure Node: Performance Improvements for the IEEE 802.6 MAN. In *Proc. of IEEE INFOCOM*, pages 636–643, San Francisco, California, 1990.
21. S. Sheu, K.A. Hua, and T.H. Hu. Virtual Batching: A New Scheduling Technique for Video-On-Demand Servers. In *Proc. of the 5th DASFAA'97*, Melbourne, Australia, April 1997.
22. S. Viswanathan and T. Imielinski. Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting. *Multimedia Systems*, 4(4):197–208, August 1996.