MPEG Video Streaming with VCR Functionality

Chia-Wen Lin, *Member, IEEE*, Jian Zhou, *Student Member, IEEE*, Jeongnam Youn, and Ming-Ting Sun, *Fellow, IEEE*

***Abstract*—With the proliferation of online multimedia content, the popularity of multimedia streaming technology, and the establishment of MPEG video coding standards, it is important to investigate how to efficiently implement an MPEG video streaming system. Digital video cassette recording (VCR) functionality (such as random access, fast forward, fast reverse, etc.) enables quick and user-friendly browsing of multimedia content, and thus is highly desirable in streaming video applications. The implementation of full VCR functionality, however, presents some technical challenges that have not yet been well resolved. In this paper, we investigate the impacts of the VCR functionality on the network traffic and the video decoder complexity. We propose a least-cost scheme for the efficient implementation of MPEG streaming video system to provide full VCR functionality over a network with minimum requirements on the network bandwidth and the decoder complexity. We also discuss our implementation of an IP-based MPEG-4 video streaming platform which provides full VCR functionality.**

***Index Terms*—Compress-domain processing, digital video cassette recording (VCR), MPEG video, streaming video, video coding.**

I. INTRODUCTION

MULTIMEDIA applications have entered an exciting era that will enormously impact our daily life. Today's multimedia technology allows network service providers to offer versatile services such as home shopping, games, video surveillance, and movie on demand [1], [2]. In these applications, video streaming technology plays an important role in media delivery. Realizing that video streaming has so many applications and so great commercial potential, many companies, organizations, and universities are developing products [3]–[7], standards, and new technologies [8] in this area.

A video streaming system should be capable of delivering concurrent video streams to a large number of users. The realization of such a system presents several challenges, such as the high storage-capacity and throughput in the video server and the high bandwidth in the network to deliver large number of video streams. With the rapid progress in processing hardware, software, storage devices, and communication networks, these problems are being solved and video streaming applications are becoming increasingly popular.

Manuscript received June 15, 2000; revised December 7, 2000. This paper was recommended by Guest Editor W. Zhu.

C.-W. Lin is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 621, Taiwan, R.O.C. (e-mail: cwlin@cs.ccu.edu.tw).

J. Zhou and M.-T. Sun are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: zhouj@ee.washington.edu; sun@ee.washington.edu).

J. Youn is with the MSL/USRL, Sony Electronics, San Jose, CA 95134 USA (e-mail: youn_98043@yahoo.com).

Publisher Item Identifier S 1051-8215(01)01854-7.

In addition to the large storage, network bandwidth, and real-time constraints, with the proliferation of online multimedia content, it is also highly desirable that multimedia streaming systems support effective and fast browsing. A key technique that enables fast and user friendly browsing of multimedia content is to provide full VCR functionality [9]. The set of effective VCR functionality includes forward, backward, stop (and return to the beginning), pause, step-forward, step-backward, fast-forward, fast-backward, and random access. This set of VCR functionality allows the users to have complete controls over the session presentation and is also useful for other applications such as video editing.

With the establishment of MPEG video coding standards [10]–[12], it is expected that many video sequences for streaming applications will be encoded in MPEG formats. However, the implementation of the full VCR functionality with the MPEG coded video is not a trivial task. MPEG video compression is based on motion compensated predictive coding with an I–B–P-frame structure. The I–B–P-frame structure allows a straightforward realization of the forward-play function, but imposes several constraints on other trick modes such as random access, backward play, fast-forward play, and fast-backward play. As will be shown later, straightforward implementation of these functions requires much higher network bandwidth and decoder complexity compared to those required for the regular forward-play function.

With the I–B–P structure, to decode a P-frame, the previously encoded I-/P-frames need to be decoded first. To decode a B-frame, both the I-/P-frames before and after this B-frame need to be first decoded. To implement a backward-play function, a straightforward implementation is for the decoder to decode the whole group of picture (GOP), store all the decoded frames in a large buffer, and play the decoded frames backward. However, this will require a huge buffer (e.g., an N-frame buffer, if the GOP size is N) in the client machine to store the decoded frames which is not desirable. Another possibility is to decode the GOP up to the current frame to be displayed, and then go back to decode the GOP again up to the next frame to be displayed. This does not require the huge buffer but will require the client machine to operate in an extremely high speed (up to N times of the normal decoding speed), which is also not desirable. The problem soon becomes impractical when the GOP size is large.

Besides the problem with backward-play, fast-forward/backward, and random-access also present difficulties. When a P-/B-frame is requested, all the related previous P-/I-frames need to be sent over the network and decoded by the decoder. This requires the network to send all the related frames besides the actually requested frame at a much higher rate which can be many times of that required by the normal forward-play. When many clients request the trick modes, it may result in much higher network traffics compared to the normal forward-play situation. It also requires high computational complexity in the client decoder to decode all these extra frames. It is possible to just send the I-frames for these trick-modes. However, if the applications use a very large GOP-size, or require high-precision in video-frame access, sending I-frames only may not be acceptable.

There are many different schemes to encode the MPEG video, depending on the desirable server/network/client complexity requirements. For example, the video can be encoded with all I-frames. This will result in the lowest complexity requirement for the client machines. However, it will require very large server storage and network bandwidth since the I-frames will result in high bit-rates. Since the network bandwidth usually is the highest concern, we assume that the video is coded with all I–B–P frames that can achieve high compression ratios for the transport over a network with minimum bandwidth resources.

Some recent works have addressed the implementation of VCR functions for MPEG compressed video for streaming video applications [1], [13]–[15]. References [1], [13], [14] address the problem of reverse-play of MPEG video streams, and [15] addresses the problem of fast-forward play. Chen *et al.* [1] described a method of transforming an MPEG I–B–P compressed bitstream into a local I–B bitstream by performing a P-to-I frame conversion to convert all the retrieved P-frames into I-frames at the client, thereby breaking the inter-frame dependencies between the P-frames and the I-frames. After the frame conversion and frame reordering, the motion-vector swapping approach developed in [13] can be used for the backward-play of the new I–B bitstream. However, this approach requires higher decoder complexity to perform the P-to-I conversion and higher storage cost to store the bit streams. Wee *et al.* [14] presented a method which divides the incoming I–B–P bitstream into two parts: I–P frames and B-frames. A transcoder is then used to convert the I–P frames into another I–P bitstream with a reversed frame order. A method of estimating the reverse motion vectors for the new I–P bitstream based on the forward motion vectors of the original I–P bitstream as described in [16] is used to reduce the computational complexity of this transcoding process. For B-frames, the motion vector swapping scheme proposed in [13] is used for the reverse-play. The transcoding process, however, still requires much computation and will cause drift due to the motion vector approximation [14]. None of the methods mentioned above fully address the problem of the extra network traffics and decoding complexity caused by the VCR functions such as fast-forward/backward and random-access. Omoigui *et al.* [15] investigated possible client–server time-compression implementations for fast-forward play and video browsing. The time compression can be implemented by storing multiple pre-encoded bitstreams with different temporal resolutions and send a bitstream with suitable temporal resolution according to the user's request. This approach does not introduce excessive network traffic but the speed-up granularity is limited by the number of pre-stored bitstreams.

In this paper, we investigate effective techniques to implement the full VCR functionality in an MPEG video streaming system. We analyze the impacts of performing VCR trick-modes on the client decoder complexity and network traffics. We propose using dual bitstreams at the server to resolve the problem of reverse play. Based on the dual-bitstream structure, we propose a novel frame-selection scheme at the server to minimize the required network bandwidth and the decoder complexity. This scheme determines the frames to stream over the network by switching between the two bitstreams based on a least-cost criterion. We present a drift-compensation scheme to eliminate the drift caused by the bitstream switching. We also describe our implementation of an MPEG-4 video streaming system supporting the full VCR functionality.

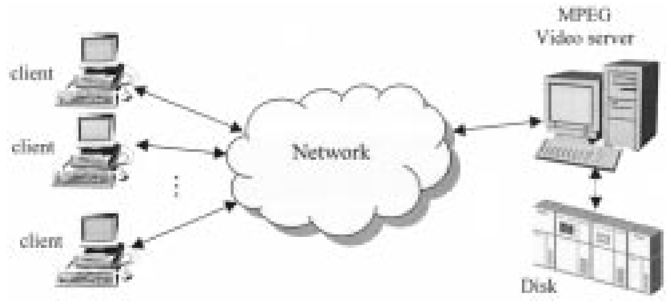


Fig. 1. MPEG video streaming.

The rest of this paper is organized as follows. In Section II, we discuss the impacts of random-access and fast-play operations on decoder complexity and network traffics. In Section III, we describe our proposed scheme for supporting full VCR functionality with least network resource and decoding effort. Section IV presents a drift-compensation scheme for the proposed least-cost bitstream switching method. In Section V, we describe our implementation of an MPEG-4 video streaming system with full VCR functionality. Finally, conclusions are given in Section VI.

II. Impacts of VCR Functionality on Decoder Complexity and Network Traffics

A block diagram of an MPEG video streaming system is shown in Fig. 1. The video streams are compressed using MPEG video coding standards and are stored in the server. The clients can view the video while the video is being streamed over the network. In each client machine, a pre-load buffer is set up to smooth out the network delay jitter. In this paper we discuss the scenario that the video is streaming over the Internet and the full VCR functionality needs to be supported. It is assumed that the applications may use a large GOP size, or require relatively high precision in video-frame access.

In the following, we provide some analyses and simulation results to show the average number of frames need to be sent through the network and decoded at the client decoder to support random-access and fast-forward play. Since the nonselected B-frames are not involved in decoding later frames and are not needed to be sent over the network or decoded by the decoder, for simplicity but without loss of generality, we focus the analyses on the cases that the bitstream contains I- and P-frames only. The results can be easily extended to the I–B–P frame structure.

A. Random Access

In the random-access operation, the decoder requests a frame with an arbitrary distance from the current displayed frame. If the requested frame is an I-frame, the server side only needs to transmit this frame, and the decoder can decode it immediately. However, if the requested frame is a P-frame, the server needs to transmit all the P-frames from the previous nearest I-frame to this requested frame.

Suppose all the GOPs in the bitstream have the same length N, and frame N_j is the random-access point

```
|<---------------N--------------->|
I  P  P  P  P  P  P  P  P  P  P  P  P  P  I
0  1     ...          N_j
```

Then, in order to decode frame N_j, frames $0, 1, \ldots, N_j - 1$ should also be sent from the server side. Assuming the random-access points are uniformly distributed, the average number of frames to be transmitted is $\overline{N}_{\text{trans}} = (N+1)/2$. For example, when N =14, $\overline{N}_{\text{trans}}$ =7.5, meaning that an average of 7.5 frames should be transmitted over the network and decoded by the decoder for the requested frame in the random-access mode.

B. Fast-Forward Play

Suppose frame N_j is the starting point of the fast-forward operation, and k is the fast-forward speed-up factor (i.e., for k =6, only one out of six frames will be displayed). Since the next frame to be displayed is N_{j+k}, the server may send the frames $N_{j+1}\ N_{j+2}\ \ldots\ N_{j+k}$, so that k frames will be received by the client side to decode the frames $N_{j+1}\ N_{j+2}\ \cdots\ N_{j+k}$ (but just displays the frame N_{j+k}).

In fact, the server may not need to transmit so many frames. For example, consider the case

```
   9                   14  15  16  17  18  19
···P̲  P  P  P  P  I   P̲   P   P   P   P...
```

where frame 9 is the current displayed frame, and frame 15 is the next frame to be displayed under the fast-forward mode ($k = 6$). Apparently, there is no need to send frames 10–13, since they are not needed for the decoding of frame 15. Therefore, the server can just send frames 14 and 15.

It is useful to derive a closed-form formula to show the impact of the fast-forward play on the decoding complexity and network traffics. One difficulty is, similar to the random-access operation, the start point of the fast-forward mode can be any frame in a GOP. However, it is reasonable to assume that the start point of a fast-forward operation is an I-frame, since we can always jump to the nearest I-frame first which will not cause unpleasant effect in viewing the video in most practical applications. Note that, with this assumption, after k/L GOPs, where $L = \gcd(k, N)$ stands for the greatest common divisor of k and N, the frame to be displayed will again be an I-frame. Therefore, the decoding pattern will repeat every k/L GOPs (i.e., $\operatorname{lcm}(k, N)$ frames, where $\operatorname{lcm}(k, N)$ is the least common multiple of k and N). We can thus derive an analytical closed-form formula based on the periodicity. In the following, we divide different combinations of N and k into three classes and derive the closed-form formula, respectively.

Case 1) $k > N,\ k \bmod N = 0$

In this case, all the P-frames are dropped, only the nonskipped I-frames are transmitted and decoded. No extra frames need to be transmitted for decoding the I-frames. Therefore, $\overline{N}_{\text{trans}} = 1$.

Case 2) $k > N,\ k \bmod N \neq 0$

As mentioned above, the decoding pattern will repeat every k/L GOPs. During each period, there are N/L frames to be requested for display. For the ith requested frame in each period ($i = 0$ *to* $N/L - 1$), a total of $(i \times k) \bmod N + 1$ (where "mod" stands for the modular operation) frames need to be transmitted and decoded. The average number of frames to be transmitted and decoding for displaying one frame is

$$\overline{N}_{\text{trans}}(k, N) = \frac{L}{N}\sum_{i=0}^{N/L-1}((i \times k) \bmod N + 1) = \frac{L}{N}\sum_{i=0}^{N/L-1}(i \times L + 1). \quad (1)$$

Case 3) $2 \leq k \leq N-1,\ N \bmod k \neq 0$

In a GOP with an I–P structure, a P-frame needs not be sent only if all its following P-frames will not be displayed at the client decoder. Therefore, in the first GOP (assuming the start point is an I-frame), the number of frames needs not be transmitted is $N \bmod k$. Similarly, the number of the P-frames which need not be sent in the jth GOP, where $1 \leq j < k/L$, is $(j \times N) \bmod k - 1$. Thus, the total number of frames that need not be transmitted in the k/L GOPs is

$$N_{\text{skip}}(k, N) = \sum_{j=1}^{k/L-1}((j \times N) \bmod k - 1) = \sum_{j=1}^{k/L-1}(j \times L - 1). \quad (2)$$

If N and k are coprime (i.e., $L = 1$), the above equation becomes

$$N_{\text{skip}}(k) = \sum_{j=1}^{k-1}(j-1). \quad (3)$$

Note that, in the case that N and k are coprime, (3) holds for any start points (not necessarily an I-frame).

In case 3, the average number of frames that need to be transmitted and decoded for displaying a requested frame can be obtained by subtracting the nontransmitted frames from the total number of frames, and then dividing it by the total number of frames to be displayed. That is

$$\overline{N}_{\text{trans}}(k, N) = \frac{\frac{k}{L} \times N - N_{\text{skip}}(k, N)}{\frac{k}{L} \times \frac{N}{k}} = k - \frac{L}{N}N_{\text{skip}}(k, N). \quad (4)$$

In summary, the average number of frames need to be transmitted and decoded for a requested frame can be expressed in closed-form as follows:

$$\overline{N}_{\text{trans}}(k, N) = \begin{cases} 1 & k = 1 \text{ or } k \bmod N = 0 \\ k - \dfrac{L}{N} \displaystyle\sum_{i=1}^{k/L-1} (i \times L - 1) & 2 \le k \le N-1 \\ \dfrac{L}{N} \displaystyle\sum_{i=0}^{N/L-1} (i \times L + 1) & k > N,\ k \bmod N \ne 0. \end{cases} \tag{5}$$

Equation (5) suggests that, if N is relatively large compared to k, $\overline{N}_{\text{trans}}$ will grow almost linearly as k increases, thereby leading to a linear increase of the decoding complexity and the network traffics.

The above analyses can be extended to the case with B-frames. The main difference from the above analyses is that, to decode a B-frame, we only need to decode the related I-/P-frames; the other B-frames do not need to be transmitted or decoded. Therefore, the number of frames that need to be transmitted and decoded for displaying one frame for GOPs with the general I–B–P structure is in general less than the I–P case. However, the analysis is very similar to the above. For simplicity of discussion, we assume that $k < N$, and divide it into two cases. Again, we assume the start point is always an I-frame.

Case 1) k is a multiple of M (where M is the distance between the I–P or P–P frames)

In this case, all the B-frames need not be transmitted and decoded. Equation (5) can be applied by replacing k and N with k/M and N/M, respectively.

Case 2) k is not a multiple of M

In this case, (2) should be modified as follows:

$$N_{\text{skip}}(k, N) = \sum_{i=1}^{k/L-1} (i \times L - 1) + N_{\text{B\_skip}} - N_{\text{P\_decodeB}} \tag{6}$$

where $N_{\text{B\_skip}}$ represents the number of B-frames that need not be decoded, and $N_{\text{P\_decodeB}}$ represents the number of P-frames need to be transmitted and decoded for decoding the B-frames in those GOPs, in which the last displayed frame is a B-frame. It is difficult to find closed-form representations for the second and third terms at the right hand side of the above equation. Computer simulations can be used to determine the numbers $N_{\text{B\_skip}}$ and $N_{\text{P\_decodeB}}$ for different combinations of N, M, and k.

Fig. 2 shows the average number of frames that need to be sent and decoded for decoding a requested frame with respect to different speed-up factors in the fast-forward operation. The test MPEG bitstream used for simulation is the "Mobile and Calendar" sequence with a length of 280 frames (20 GOPs in our example), which was encoded at 3 Mbits/s with a frame-rate of 30 fps with an I–P structure. The start points are randomly generated. Fig. 3 depicts the average bit-rates required for sending the "Mobile and Calendar" video stream with respect to different speed-up factors. From the above analysis, in the fast-forward/backward and random-access operations, the server needs to send several extra frames to the decoder to display one frame, thereby resulting in a heavy burden on the network (especially when the number of users is large) and increasing the decoder complexity.

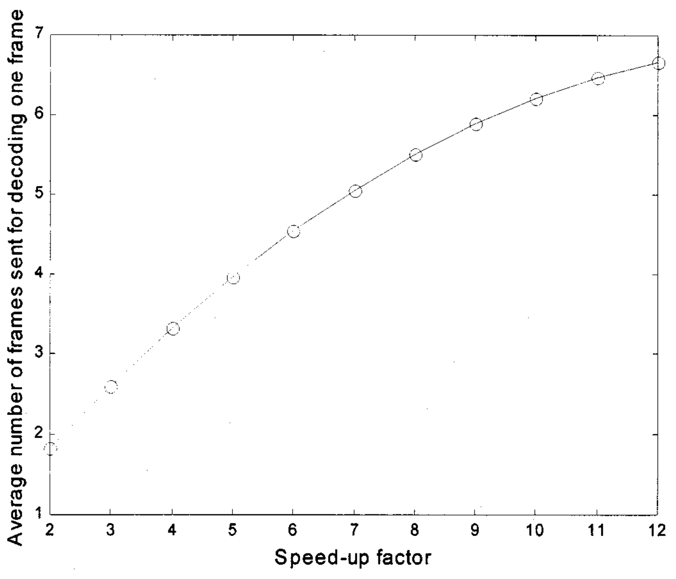


Fig. 2. Average number of frames that need to be sent for decoding a frame with respect to different speed-up factors in fast-forward play.

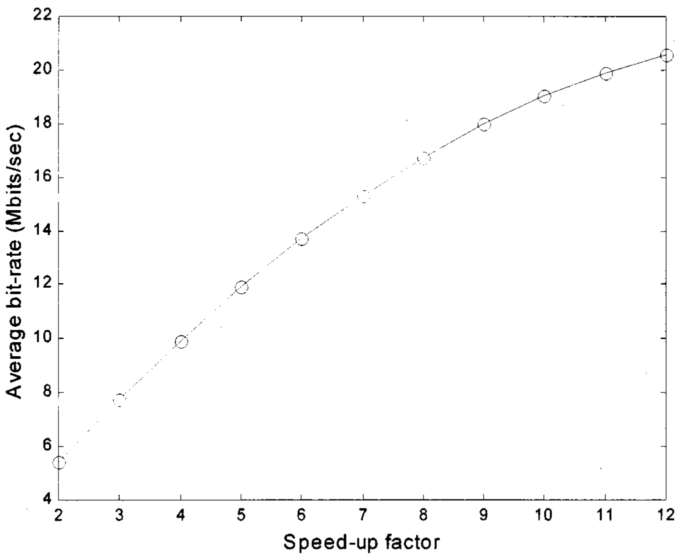


Fig. 3. Average bit-rates for sending the "Mobile and Calendar" sequence over network with respect to different speed-up factors in fast-forward play.

III. Supporting Full VCR Functionality with Minimal Network Bandwidth and Decoder Effort

A. Dual Bitstreams with Least-Cost Frame Selection

To solve the problem of the backward-play operation, we propose to add a reverse-encoded bitstream in the server [20], i.e., in the encoding process, after we finish the encoding and reach the last frame of the video sequence, we encode the video frames in the reverse order to generate a reverse-encoded bitstream. If the server only has the forward bitstream (i.e., the original sequence is unavailable), we can decode the forward bitstream up to two GOPs each time in the reverse direction (i.e., from the last GOP to the first GOP) then re-encode the video in the reverse order. The generation of the reverse bitstream is done off-line. For simplicity of the presentation, in this paper, we use an example in which the video is coded in I-/P-frames with a GOP size of 14 frames, as is shown below. The extension of our discussion to the case with the general I–B–P GOP structure is straightforward.

In the diagram shown at the bottom of the page, we arrange the encoding so that the I-frames in the reverse bitstream are interleaved between I-frames in the forward bitstream. In this way, the required number of frames sent by the server and decoded by the decoder can be further reduced as will be explained later. Alternatively, the I-frames in both streams can be aligned to save storage, since the two I-frames in the forward and reverse bitstreams are the same, and only need to be stored once. Two metadata files recording the location of the frames in each compressed bitstream are also generated so that the server can switch from the forward-encoded bitstream to the reverse-encoded bitstream and vice-versa easily. I-frames represent the points of access to decode the sequence from any arbitrary position. With the reverse-encoded bitstream, when the client requests the backward-play mode, the server will stream the bits from the reverse-encoded bitstream. Using this scheme, the complexity of the client machine and the required network bandwidth for the backward-play mode can be minimized. The storage requirement of the server will be about doubled. However, this is usually much more desirable than to require a large network bandwidth and to increase the complexity of the client machine since the network bandwidth is more precious and there may be a large number of client machines in the streaming video applications. Since the encoding of the video is done off-line and can be automated, the extra time needed in producing the reverse encoded bitstream is not an important concern.

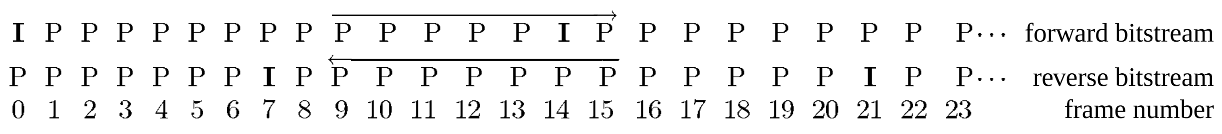
To reduce the decoding complexity and the network traffics in the fast-forward/backward and the random access modes, we propose a frame-selection scheme which minimizes a predefined "cost" using bitstream switching. The cost can be the decoding effort at the client decoder or the traffic over the networks, or a combination of both. This is further explained as follows.

Let $c_{\rm R\_C}$ stand for the cost of decoding the next requested P-frame from the current displayed frame, $c_{\rm R\_FI}$ stand for the cost of decoding the next requested P-frame from the closest I-frame in the forward bitstream, and $c_{\rm R\_RI}$ stand for the cost of decoding the next requested frame from the closest I-frame of the reverse-encoded bitstream. To minimize the number of frames sent to the decoder, the costs can be the distances from the possible reference frames to the next requested frame. To minimize the network traffics, the costs can be the numbers of bits from the possible reference frames required for decoding the next requested frame. The bit-rate calculation can be done simply by recording the number of bits used for each encoded frame in the metadata file in the pre-encoding process, and summing up the bit-rates of those frames to be sent. In general, a larger number of frames to be sent implies heavier network load. However, it also depends on the numbers of I-, P-, and B-frames to be sent since the numbers of bits produced by these three types of frames vary greatly. It is also possible to use different weights to combine the two costs according to the channel condition and the client capability. Based on the current play-direction, the requested mode, and the costs $c_{\rm R\_C}$, $c_{\rm R\_FI}$, and $c_{\rm R\_RI}$, the reference frame to the next requested frame with the least cost will be chosen to initiate the decoding. This will also determine the selection of the next bitstream and the decoding direction. This least-cost criterion will only be activated in the fast-forward/backward and the random access modes to avoid frequent bitstream switching in the normal forward/backward operations.

To illustrate the scheme, we use the example in Section II again, assuming that the previous mode was backward-playing and the requested mode is fast-backward with a speed-up factor of 6, which needs to display a sequence of frame numbers 20, 14, 8, 2. For simplicity, in the following examples shown at the bottom of the next page, we use the minimum decoding distance criterion to illustrate the selection of the next reference frame and the effectiveness of the proposed method.

The algorithm will operate as follows.

1) The current position is frame 20, which was decoded using the reverse bitstream (R).
2) Frame 14 will be decoded from the forward bitstream (F) directly since it is an I-frame.
3) Frame 8 will be decoded from frame 7 of the backward bitstream, since the distance between frame 7 of the reverse bitstream (an I-frame) and the requested frame (frame 8) is less than the distances between the requested frame and the current decoded frame (frame 14 of the reverse bitstream), and the closest I-frame of the forward bitstream (it's also frame 14). Note that in this case, we use frame 7 of the reverse bitstream (an I-frame) as an approximation of frame 7 of the forward bitstream (a P-frame) to predict frame 8 of the forward bitstream. This will cause some drift. However, in the fast-forward/backward modes, the drift is relatively insensitive to human eyes due to the fast change of the content displayed. Also, any I-frame in the play will terminate


I P P P P P P P P P P P P P I P P P P P P P P P··· forward bitstream
P P P P P P P I P P P P P P P P P P P P P I P P··· reverse bitstream
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 frame number

the drift. The drift problem will be further investigated in the next section.

4) Frame 2 will be decoded from frames 0 and 1, using the forward bitstream, since the decoding effort from frame 0 of the forward bitstream (an I-frame) is the minimum.

The bitstream sent from the server will have the following form:

P	**I**	I	**P**	I	P	**P**$\cdots$	frame type
20	**14**	7	**8**	0	1	**2**$\ldots$	frame number
R	***F***	*R*	***F***	*F*	*F*	***F***$\ldots$	selected bitstream

The frames indicated by the bold-face are those to be displayed at the client side. In this way, we only need to send and decode 6 frames. Without the minimum effort decoding scheme, we will need to send and decode 13 frames from the reverse bitstream.

In the case of random access, frame skipping will be performed followed by normal forward-play. For example, the client requests random access to frame 22 when the current decoded frame is frame 3. With the proposed method using the minimum decoding distance criterion, the server streams the bitstream as follows:

P	I	**P**	**P**	**P**$\cdots$	frame type
3	21	**22**	**23**	**24**$\cdots$	frame number
F	*R*	***F***	***F***	***F***$\cdots$	selected bitstream

In this example, we only need to send and decode two frames to reach frame 22. Without our proposed least-cost scheme, it will require to send and decode nine frames from frame 14 (an I-frame) using the forward bitstream. Again, in this example, for frame 21, we use the I-frame in the reverse bitstream to approximate the P-frame in the forward bitstream. This will cause drift but the drift will only last a few frames within the GOP (a fraction of a second) since the video content will be refreshed by the I-frame in the next GOP. Thus, it should not be a problem.

If the minimum decoding distance criterion is used (i.e., to minimize the number of frames sent to the decoder), the proposed scheme will guarantee that the maximum amount of decoding to access any frame in the sequence is less than $N/4$ frames if the I-frames in the forward and the reverse bitstreams are interleaved. In addition, no large temporary buffer is required in the decoder. If the I-frames in the forward and the reverse bitstreams are aligned, the maximum amount of decoding to access any frame will be less then $N/2$ frames.

B. Performance Analysis of the Proposed Dual-Bitstream Least-Cost Method

In the following, we analyze the performance of the proposed method using the minimum decoding distance criterion for the random-access and the fast-forward/backward modes.

1) Random Access: A typical structure of the system with dual bitstreams is shown in the second example at the bottom of the page, where both the forward bitstream and the reverse bitstream have the same GOP structure with the length of N. As in the analysis in Section II, we label the frames in the GOP, where the requested frame lies as $0, 1, 2, \ldots, N-1$, frame N_j is the random access point. $N_{\rm RI}$ is the position of the I-frame in the reverse bitstream.

In this case, the frames to be transmitted for decoding frame N_j will be decided by two distance measures, one is the distance from frame N_j to the nearest I frame in the forward bitstream, and the other distance is from N_j to the nearest I-frame in the reverse bitstream.

Assume $N_j \in [0, N-1]$, $N_{\rm RI}$ is in the range of $[1, N-1]$. For simplicity but without loss of generality, we assume that N is even and $N_{\rm RI}$ is odd as in our previous example. We can observe that:

1) the minimum number of total frames to be sent over the network is 1 (when $N_j = 0$ or $N_j = N_{\rm RI}$);
2) the maximum number of total frames to be sent over the network is

$$\max\left(\frac{N_{\rm RI}-1}{2}+1,\ \frac{N-1-N_{\rm RI}}{2}+2\right);$$

Frame No.		0	1	**2**	3	4	5	6	7	**8**	9	10	11	12	13	**14**	15	16	17	18	19	**20**	21	22	23
F	$->$	**I**	P	P	P	P	P	P	P	P	P	P	P	P	P	**I**	P	P	P	P	P	P	P	P	P$\cdots$
R	$<-$	P	P	P	P	P	P	P	**I**	P	P	P	P	P	P	P	P	P	P	P	P	P	**I**	P	P$\ldots$

Frame No.	0	1	**2**	3	4	5	6	7	**8**	9	10	11	12	13	**14**	15	16	17	18	19	**20**	21	22	23
						$\longrightarrow$							decoding direction											
F	**I**	P	P	P	P	P	P	P	P	P	P	P	P	P	**I**	P	P	P	P	P	P	P	P	P$\ldots$
						$\longleftarrow$							decoding direction											
R	P	P	P	P	P	P	P	**I**	P	P	P	P	P	P	P	P	P	P	P	P	P	**I**	P	P$\ldots$
				N_j				$N_{\rm RI}$																

3) the average number of total frames to be sent over the network is

$$\begin{aligned}\overline{N}_{\text{trans}} &= 1 + \sum_{i=0}^{(N_{\text{RI}}-1)/2} \frac{i}{N} + \sum_{i=(N_{\text{RI}}+1)/2}^{N_{\text{RI}}} \frac{(N_{\text{RI}}-i)}{N} \\ &\quad + \sum_{i=N_{\text{RI}}+1}^{(N_{\text{RI}}-1+N)/2} \frac{(i-N_{\text{RI}})}{N} + \sum_{i=(N_{\text{RI}}+1+N)/2}^{N-1} \frac{(N-i)}{N} \\ &= 1 + \frac{2N_{\text{RI}}^2 - 2NN_{\text{RI}} + N^2 - 2}{4N}.\end{aligned}$$

By taking the derivative with respect to N_{RI}, we can find that when N_{RI} takes the odd number closest to $N/2$, $\overline{N}_{\text{trans}}$ can take the minimum value of

$$\overline{N}_{\text{trans}} = \begin{cases} 1 + \dfrac{N}{8} - \dfrac{1}{2N}, & \left(\dfrac{N}{2} = \text{odd}\right) \\ 1 + \dfrac{N}{8}, & \left(\dfrac{N}{2} = \text{even}\right). \end{cases}$$

For example, when $N = 14$, $N_{\text{RI}} = 7$, $\overline{N}_{\text{trans}} = 2.71$, meaning that an average of 2.71 frames should be transmitted to decode one requested frame in the random access mode. Apparently, this is much better than the case in Section II (7.5 frames without our scheme).

2) Fast Forward-Play: In the proposed method, when the speed-up factor k is larger than $N/4$, the server always can find an I-frame in one of the two bitstreams which has a shorter distance to the next displayed frame from the current displayed P-frame, since the distance for the nearest I-frame is guaranteed to be equal to or less than $N/4$. In this case, the number of frames to be sent for displaying a requested frame will have a range of $(1, N/4+1)$. It is difficult to derive a closed-form formula to calculate the average number of frames transmitted and decoded for each requested frame for general N and k cases with any start points. For simplicity of analysis, in the following, we assume the start point is an I-frame in the forward bitstream and only consider the case that $k > N/4$.

Since the start point is an I-frame, the requested frames will again become I-frames every k/L GOPs, meaning that the decoding pattern will repeat every k/L GOPs. The distance between the ith requested frame ($i = 0, 1, \ldots$) and its proceeding I-frame is $i \times k \bmod N$, which is a multiple of L. No distances will be the same in a period, otherwise we can find a shorter period and it's a contradiction. Therefore we can conclude that, in a period of k/L GOPs, there are a total of N/L requested frames with equally spaced distances, say $0, L, 2\times L, \ldots, (N/L-1)\times L$ frames away from their nearest proceeding I-frames. Based on this property, we can derive closed-form formulas to calculate the average number of frames transmitted for decoding a requested frame. We divide the analysis into two classes.

Case 1) N is even

In this case, every $N/2$ frames there is an I-frame (in either the forward or the backward bitstream, when the I-frames are interleaved) which can be used as an anchor frame to initiate the decoding of the requested frames. The average number of frames transmitted for decoding a requested frame is

$$\begin{aligned}\overline{N}_{\text{trans}} &= 1 + \frac{1}{\frac{N}{L}} \left(\sum_{i=0}^{\lfloor N/4L \rfloor} i \times L + \sum_{i=\lfloor N/4L \rfloor+1}^{\lfloor N/2L \rfloor} \left(\frac{N}{2} - i \times L\right) \right. \\ &\quad + \sum_{i=\lfloor N/2L \rfloor+1}^{\lfloor 3N/4L \rfloor} \left(i \times L - \frac{N}{2}\right) \\ &\quad \left. + \sum_{i=\lfloor 3N/4L \rfloor+1}^{N/L-1} (N - i \times L) \right) \\ &= 1 + \frac{2L}{N} \left(\sum_{i=0}^{\lfloor N/4L \rfloor} i \times L + \sum_{i=\lfloor N/4L \rfloor+1}^{\lfloor N/2L \rfloor} \left(\frac{N}{2} - i \times L\right) \right)\end{aligned} \tag{7}$$

where $\lfloor X \rfloor$ represent the largest integer smaller than X. It is interesting to note that, when N and k are coprime (i.e., $L = 1$), $\overline{N}_{\text{trans}}$ is only a function of N regardless of the values of k. The above equation becomes

$$\begin{aligned}\overline{N}_{\text{trans}} &= 1 + \frac{2}{N} \left(\sum_{i=0}^{\lfloor N/4 \rfloor} i + \sum_{i=\lfloor N/4 \rfloor+1}^{N/2} \left(\frac{N}{2} - i\right) \right) \\ &= \begin{cases} 1 + \dfrac{N}{8}, & \dfrac{N}{2} \text{ is even} \\ 1 + \dfrac{N}{8} - \dfrac{1}{2N}, & \dfrac{N}{2} \text{ is odd.} \end{cases}\end{aligned} \tag{8}$$

In fact, for the cases that N and k are not coprime, the results of (7) and (8) are still very close. Therefore, the simple formula in (8) can be applied in most of the cases that N is even.

Case 2) N is odd

Similar to the above derivation, we can obtain the following formula:

$$\begin{aligned}\overline{N}_{\text{trans}} &= 1 + \frac{1}{\frac{N}{L}} \left(\sum_{i=0}^{\lfloor (N-1)/4L \rfloor} i \times L \right. \\ &\quad + \sum_{i=\lfloor (N-1)/4L \rfloor+1}^{\lfloor (N-1)/2L \rfloor} \left(\frac{N-1}{2} - i \times L\right) \\ &\quad + \sum_{i=\lfloor (N-1)/2L \rfloor+1}^{\lfloor (3(N-1))/4L \rfloor} \left(i \times L - \frac{N-1}{2}\right) \\ &\quad \left. + \sum_{i=\lfloor (3(N-1))/4L \rfloor+1}^{N/L-1} (N - i \times L) \right).\end{aligned} \tag{9}$$

When N and k are coprime, the above equation becomes

$$\overline{N}_{\text{trans}} = 1 + \frac{N}{8} - \frac{1}{8N}. \tag{10}$$

We have simulated the situation of the I–P structure for $N = 14$ with a number of randomly generated start points. Two bitstreams generated by forward and reverse encoding the 280-frame "Mobile and Calendar" test sequence at 3 Mbits/s with the frame rate of 30 fps with an I–P structure are used for the simulation. Fig. 4 shows the comparison of the average number of the frames transmitted to the decoder for decoding a requested frame with and without the proposed dual-bitstream least-cost method with respect to different speed-up factors in the fast-forward operation. The simulation result is very close to the value 2.71 calculated by using (8) with $N = 14$ when the speed-up factor $k > N/4$. The fast-backward play case will also have similar result. Fig. 5 depicts the comparison of the average bit-rates required to send the video stream with respect to different speed-up factors. Note that, with the proposed method, when the speed-up factor reaches around $N/4$ (e.g., 3.5 in our example), the decoding complexity and the network traffic will not continue to grow even when the speed-up factor gets higher. Compared to the results in Section II, it is obvious that the proposed method can achieve significant performance improvement in terms of the decoder complexity and the network traffic load. When the speed-up factor $k \geq N/4$, the proposed method guarantees a nearly constant decoding and network traffic cost.

IV. Drift Compensation

As mentioned above, in the proposed scheme, I- or P-frames of one bitstream may be used to approximate P-frames of the other bitstream. This approximation, however, will lead to frame mismatch and thus cause drift when the approximated frames are used as the reference frames to predict the following P-/B-frames as illustrated in Fig. 6. In Fig. 6, the "Mobile and Calendar" sequence is encoded at a fixed quantization scale ($Q = 16$) and a fixed bit-rate (3 Mbits/s), respectively. The GOP size is 14 and the speed-up factor is 6. As shown in Fig. 6, when the server performs an I-to-P or a P-to-P approximation by using the proposed bitstream switching, there is a PSNR drop. Fig. 6 suggests that the drift caused by the bitstream switching can be as large as 2.5 dB and will last until the next I-frame. However, the subjective degradation observed is not significant, since the fast display speed in the fast forward/backward modes will mask most of the spatial distortions.

In the random access mode, the drift will only last a few frames within a GOP, and thus will not cause serious degradation. In the fast-forward/backward mode, the drift is relatively insensitive to human eyes due to the fast changes of the content displayed. However, in some applications, it may still be desirable to prevent the drift. The drift problem can be resolved by adding two bitstreams consist of all P-frames for the drift-compensated bitstream switching. This is further explained using the example shown at the bottom of the next page, where D^{FR} is a bitstream used for switching from the I- or P-frames of the forward bitstream to the P-frames of the reverse bitstream, while D^{RF} is used for switching from the I- or P-frames of the reverse bitstream to the P-frames of the forward bitstream. The bitstream D^{FR} is obtained as follows:

$$D_n^{\mathrm{FR}} = \mathrm{Pred}(F_n, R_{n-1}) \tag{11}$$

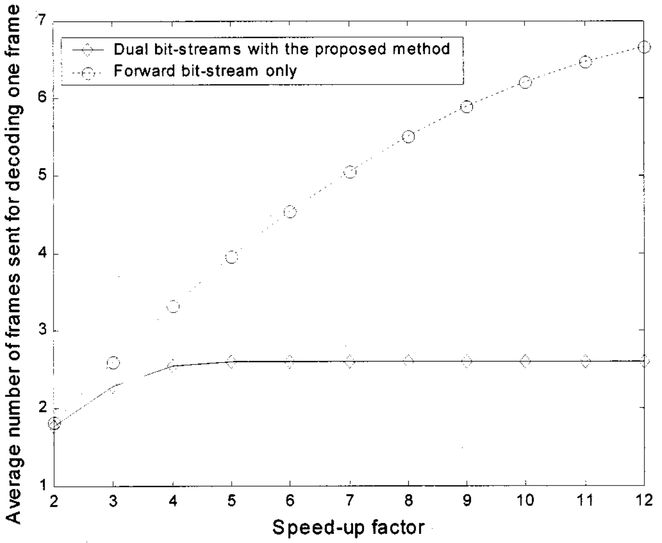


Fig. 4. Estimated number of frames to be sent for decoding a frame using the proposed method with respect to different speed-up factors.

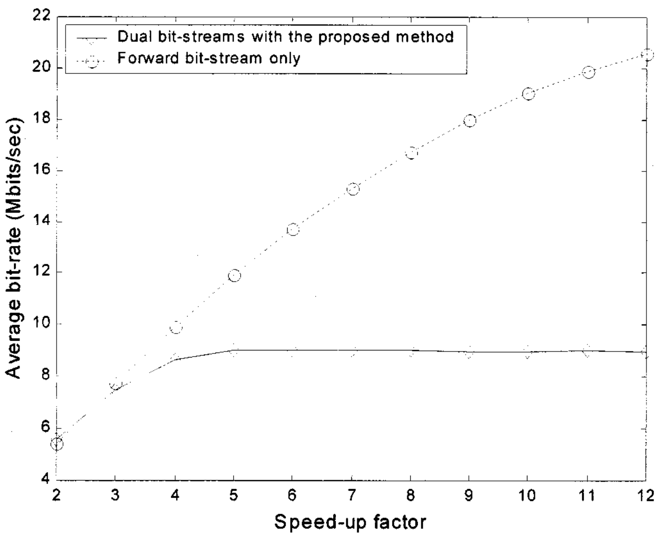


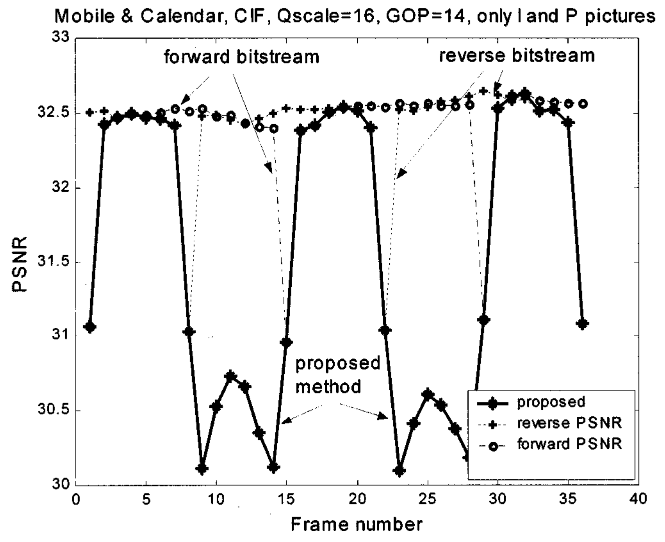
Fig. 5. Average bit-rates to send the "Mobile and Calendar" sequence over network using the proposed method with respect to different speed-up factors.

and

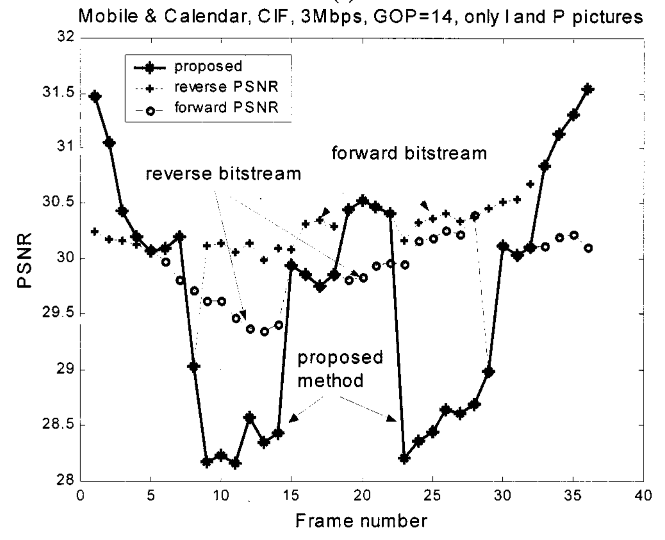
$$D_n^{\mathrm{RF}} = \mathrm{Pred}(R_n, F_{n+1}) \tag{12}$$

where $\mathrm{Pred}(A, B)$ represents an inter-frame prediction process that frame B is predicted from the reference frame A. When performing the bitstream switching, the correctly predicted frame is used for switching between the forward and the reverse bitstreams. For example, if the bitstream is switched from F_n (an I- or P-frame) to R_{n-1} (a P-frame), then the server will send the frames as $\ldots, F_n, D_n^{\mathrm{FR}}, R_{n-2}, \ldots$, instead of sending $\ldots, F_n, R_{n-1}, R_{n-2}, \ldots$. With the two drift correction bitstreams, the proposed method will generate a bitstream for the fast-reverse example in Section III-A as follows:

$$\begin{array}{cccccccl}
\mathbf{P} & \mathbf{I} & \mathrm{I} & \mathbf{P} & \mathrm{I} & \mathrm{P} & \mathbf{P}\ldots & \text{frame type} \\
\mathbf{20} & \mathbf{14} & 7 & \mathbf{8} & 0 & 1 & \mathbf{2}\ldots & \text{frame number} \\
\boldsymbol{R} & \boldsymbol{F} & R & \boldsymbol{D}^{\mathbf{RF}} & F & F & \boldsymbol{F}\ldots & \text{selected bitstream}
\end{array}$$



(a)



(b)

Fig. 6. PSNR comparison of the forward bitstream, the reverse bitstream, and the bitstream generated using the proposed method in the fast-forward mode for the "Mobile and Calendar" sequence, the GOP size is 14, and the speed-up factor is 6. (a) Sequence is quantized at a fixed quantization scale $Q = 16$. (b) Sequence is encoded at 3 Mbits/s.

Since D^{RF} is the encoded based on the decoded frames from the forward and the reverse bitstreams, the drift can be compensated very well. If the prediction errors of the drift-compensated predictive frames in D^{RF} and D^{FR} are losslessly encoded, there will be no drift. Otherwise, there will be small drift. The drift will depend on the quantization step-size used in the encoding. A finer quantizer will lead to lower drift, while increasing the storage for the drift compensation bitstreams. Since the encoding process to obtain all the bitstreams is done off-line in streaming video applications, the encoding complexity is not a major concern.

It should be noted that if the I-frames of the two bitstreams are interleaved, and the speed-up factor is high enough (e.g., the frame-skipping distance $\geq N/4$), in the proposed method, only replacing P-frames with I-frames will be sufficient, because we always can find an I-frame in one of the two bitstreams which has shorter distance to the next requested frame than the current decoded P-frame. In this case, we only need to store the drift compensation frames for all the I-frames of both bitstreams. In the fast-forward/backward operations with small speed-up factors (e.g., 2 or 3); however, the proposed least-cost scheme has limited gain on the decoding complexity and the network traffics as shown in Figs. 4 and 5. Thus, a possible low-complexity solution for the fast-forward/reverse play is

```
if $k < N/4$
        use dual bitstreams without performing
            bitstream switching
else
        use bitstream switching with I −> P
            drift-compensation only.
```

Using this modified scheme, only the drift-compensations for the I −> P frames need to be created, thus the storage cost for the drift compensation frames can be reduced drastically without significant performance sacrifice in typical MPEG applications.

V. Implementation of an MPEG-4 Video Streaming System with Full VCR Functionality

We have implemented an MPEG-4 [20] video streaming system to demonstrate the effectiveness of our scheme. Fig. 7 illustrates the overall structure of the system and depicts its major functional blocks.

The client station is connected to the remote video server over an IP network and requests access to a compressed video sequence. Two logical channels are established between the server and the client: the data channel and the control channel. The

Frame No.	0	1	**2**	3	4	5	6	7	**8**	9	10	11	12	13	**14**	15	16	17	18	19	**20**	21	22	23
						$\longrightarrow$							decoding direction											
D^{FR}	**P**	P	P	P	P	P	P	P	P	P	P	P	P	P	**P**	P	P	P	P	P	P	P	P	P
F	**I**	P	P	P	P	P	P	P	P	P	P	P	P	P	**I**	P	P	P	P	P	P	P	P	P...
						$\longleftarrow$							decoding direction											
R	P	P	P	P	P	P	P	**I**	P	P	P	P	P	P	P	P	P	P	P	P	P	**I**	P	P...
D^{RF}	P	P	P	P	P	P	P	**P**	P	P	P	P	P	P	P	P	P	P	P	P	P	**P**	P	P

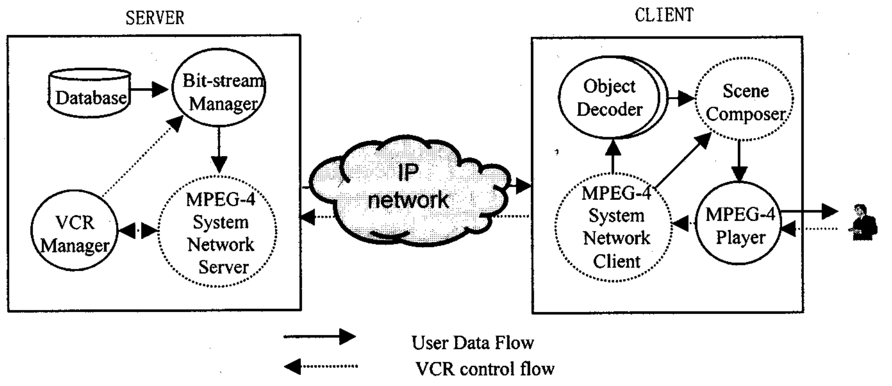


Fig. 7. System architecture of the proposed MPEG-4 video streaming system The MPEG-4 Player provides the user interface and displays video frames sent by the server according to the user's requests. In Fig. 8, we show a screen shot of the MPEG-4 Player, which illustrates the user interface for the full VCR functionality.

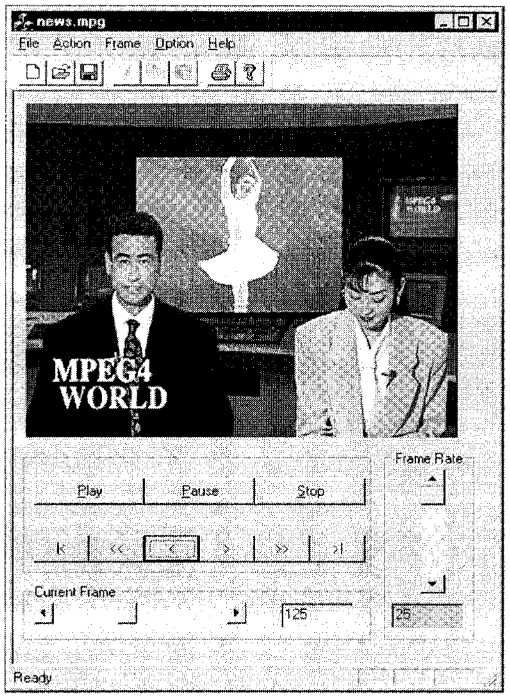


Fig. 8. MPEG-4 Player.

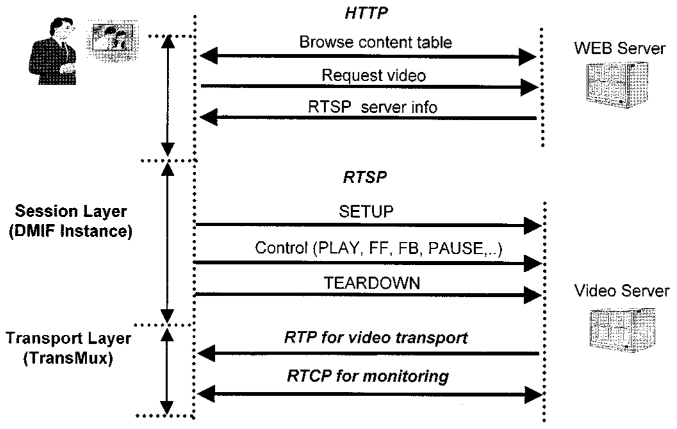


Fig. 9. Protocol framework of our system.

server delivers the requested MPEG-4 bitstream through the data channel and receives VCR commands through the control channel.

As shown in Fig. 7, the video server consists of a VCR Manager, a Bitstream Manager, an MPEG-4 System Network Server, and a Video Database. The client station consists of an MPEG-4 Player, an MPEG-4 Object Decoder, a Scene Composer, and an MPEG-4 System Network Client. The client station will access the video server and interactively retrieve a video sequence through the Graphic User Interface of the MPEG-4 Player.

According to the specific VCR function that the user selected through the user interface, the Player generates the requested frame-number and sends it to the MPEG-4 System Network Client. The MPEG-4 Object Decoder receives the corresponding bitstream from the MPEG-4 System Network Client, performs the decoding procedure of the received bitstream, and transfers the decoded frames to the MPEG-4 Player (see Fig. 8).

The MPEG-4 System Network Server manages the network connection. The Bitstream Manager maintains the record of the video bitstreams, and the VCR Manager handles the frame-number generation for the frame to be transmitted over the network.

In our protocol implementation, we use the Real Time Streaming Protocol (RTSP) [17], which is a client–server application-level protocol for controlling the delivery of data with real-time properties. It establishes and controls time-synchronized streams of continuous media such as audio and video. It uses transport protocols such as UDP, multicast UDP, TCP, and Real-time Transport Protocol (RTP) to deliver the continuous streams. The video data is delivered using the RTP [18], [19]. This has been implemented as an instance of the DMIF in the MPEG-4 Systems. The protocol framework of the implemented MPEG-4 streaming system is shown in Fig. 9.

VI. Conclusion

In this paper, we discussed issues in implementing an MPEG video streaming system with full VCR functionality. We showed that when the users request reverse-play, fast-forward/reverse-play, or random-access, it may result in much higher network traffics than the normal-play mode. These trick modes may also require high client machine complexity. We proposed to use a reverse-encoded bitstream to simplify the client terminal complexity while maintaining the low network bandwidth requirement. We proposed a minimum-cost frame-selection scheme which can minimize the number of frames needed to be sent over the network and to be decoded. We proposed a drift-compensation scheme to limit the drift. We also described our implementation of an MPEG-4 video streaming system. We showed that with our proposed scheme, an MPEG-4 video streaming system with full VCR functionality can be implemented to minimize the required network bandwidth and decoder complexity.

Acknowledgment

The authors would like to thank J. Xin for providing some simulation results, and the anonymous reviewers for their helpful comments, especially on the derivation of the closed-form formula for the fast forward-play analyses.

References

[1] M. S. Chen and D. D. Kandlur, "Downloading and stream conversion: Supporting interactive playout of videos in a client station," in *Proc. 2nd Int. IEEE Conf. Multimedia Computing and Systems*, 1995, pp. 73–80.

[2] T. D. C. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, vol. 13, pp. 14–24, Aug. 1994.

[3] Microsoft Windows Media, Microsoft Corporation Inc. [Online]. Available: http://www.microsoft.com/windows/windowsmedia/

[4] Apple QuickTime Player, Apple Corporation Inc. [Online]. Available: http://www.apple.com/quictime/

[5] Real Networks RealPlayer [Online]. Available: http://www.real.com/

[6] Relay Networks ReplayTV [Online]. Available: http://www.replay.com/

[7] TiVo Inc [Online]. Available: http://www.tivo.com/

[8] D. Wu, Y. T. Hou, and Y.-Q. Zhang, "Transporting real-time video over the Internet: Challenges and approaches," *Proc. IEEE*, vol. 88, Dec. 2000.

[9] F. C. Li, A. Gupta, E. Sanocki, L. He, and Y. Rui, "Browsing digital video," Microsoft Research, Tech. Rep. MSR-TR-99-67, [online]. Available:ftp://ftp.research.microsoft.com/pub/tr/tr-99-67.pdf, Sept. 1999.

[10] *Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbits/s*, (MPEG-1), Oct. 1993.

[11] *Generic Coding of Moving Pictures and Associated Audio*, ISO/IEC 13 818-2, (MPEG-2), Nov. 1993.

[12] *Coding of Moving Pictures and Associated Audio MPEG98/W2194*, (MPEG-4), Mar. 1998.

[13] S. Cen, "Reverse playback of MPEG video," U.S. Patent 5 739 862, Apr. 14, 1998.

[14] S. J. Wee and B. Vasudev, "Compressed-domain reverse play of MPEG video streams," in *Proc. SPIE Conf. Multimedia Systems and Applications*, Nov. 1998, pp. 237–248.

[15] N. Omoigui, L. He, A. Gupta, J. Grudin, and E. Sanocki, "Time-compression: System concerns, usage, and benefits," in *Proc. ACM SIGHI Conf.*, May 1999, pp. 136–143.

[16] S. J. Wee, "Reversing motion vector fields," in *Proc. IEEE Int. Conf. Image Processing*, Oct. 1998, pp. 209–212.

[17] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," Internet Engineering task Force, RFC 2326, Apr. 1998.

[18] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-ime applications," Internet Engineering Task Force, RFC 1889, Jan. 1996.

[19] H. Schulzrinne, D. Hoffman, M. Speer, R. Civanlar, A. Basso, V. Balbanian, and C. Herpel, "RTP payload format for MPEG-4 elementary streams," Internet Engineering Task Force, Internet Draft, Mar. 1998.

[20] C.-W. Lin, J. Youn, J. Zhou, M.-T. Sun, and I. Sodagar, "MPEG video streaming with VCR functionality," in *Proc. IEEE Int. Symp. Multimedia Software Eng.*, Taipei, Taiwan, Dec. 2000.

Chia-Wen Lin (S'94–M'00) received the M.S. and Ph.D. degrees in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 1992 and 2000, respectively.

He joined the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan, in August 2000, where he is currently an Assistant Professor. Before that, he was a Section Manager of the CPE and Access Technologies Department at the Computer and Communications Research Laboratories, Industrial Technology Research Institute (CCL/ITRI), Taiwan. During April to August 2000, he was with the Information Processing Lab, Department of Electrical Engineering, University of Washington at Seattle, as a Visiting Scholar. He has six patents pending and more than 30 technical papers published. His research interests include video coding, multimedia technologies, and digital transmission systems design.

Dr. Lin received a Research Achievement Award from ITRI in 2000.

Jian Zhou (S'00) received the B.S. and M.S. degrees in 1996 and 1999, respectively, both from the Department of Electronic Engineering, Tsinghua University, Beijing, China. He is currently working toward the Ph.D. degree in the Information Processing Lab, Department of Electrical Engineering, University of Washington at Seattle.

His research interests include video coding and multimedia applications over network.

Jeongnam Youn received the B.S. degree from Hanyang University, Korea, in 1988, the M.S. degree from Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea, in 1990, and the Ph.D. degree from the University of Washington at Seattle in 2000, all in electrical engineering.

He is with the Sony U.S. Research Lab for Video Algorithms and Multimedia Systems Developments, San Jose, CA. Previously, he had been with Equator Technologies, Inc. as a Video Scientist, where he developed video processing algorithms for multimedia processors. During 1990–1995, he was a Member of Technical Staff with Korea Telecom. His research interests are video signal processing, multimedia networking, video transcoding, and multimedia processor applications.

Ming-Ting Sun (S'79–M'81–SM'89–F'96) received the B.S. degree from National Taiwan University in 1976, the M.S. degree from University of Texas at Arlington in 1981, and the Ph.D. degree from University of California, Los Angeles in 1985, all in electrical engineering.

He joined the University of Washington at Seattle in August 1996, where he is currently a Professor. Previously, he was the Director of the Video Signal Processing Research Group at Bellcore. He has been awarded seven patents and has published more than 100 technical papers, including ten book chapters in the area of video technology.

Dr. Sun is the Editor-in-Chief of IEEE TRANSACTIONS ON MULTIMEDIA and was the Editor-in-Chief of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT) from 1995 to 1997. From 1988 to 1991, he served as the Chairman of the IEEE Circuits and Systems (CAS) Standards Committee and established an IEEE Inverse Discrete Cosine Transform Standard. He was also a general Co-Chair of the Visual Communications and Image Processing 2000 Conference. He received a Golden Jubilee Medal from the IEEE CAS Society in 2000, was co-recipient of the TCSVT Best Paper Award in 1993, and received an Award of Excellence from Bellcore in 1987 for the work on Digital Subscriber Line.