DIVeR : A Dynamic Interactive Video Retrieval Protocol for disk array based servers

Senthil Sengodan
Communication System Lab.
Nokia Research Center
3 Burlington Woods Drive, Ste. 260
Burlington, MA 01803, U.S.A
senthil.sengodan@research.nokia.com

Victor O. K. Li
Dept. of Electrical & Electronic Engg.
The University of Hong Kong
Room 601, Chow Yei Ching Bldg.
Pokfulam Road, Hong Kong
vli@eee.hku.hk

Abstract

Video-on-demand (VOD) is a very promising multimedia application of the near future. In order for such a service to be commercially viable, efficient storage and retrieval schemes need to be designed. Grouping MPEG frames into segments and retrieving segments instead of individual frames can result in a more efficient use of disk retrieval resources. In conjunction with efficient scheduling protocols for handling multiple users, a large number of users can be supported resulting in a more cost-effective system with good Quality of Service (QoS). Our contribution in this paper is two-fold:

- *we propose a scheme for grouping MPEG frames into segments wherein no frames are discarded during fast playback (with segment skipping) due to the unavailability of other frames needed for their decoding. The grouping scheme can handle any desired segment size and the fast playback rate can also take on any desired value.*
- *we propose the Dynamic Interactive Video Retrieval (DIVeR) protocol, a protocol for scheduling the retrieval of multiple users from disk-array servers. DIVeR has a low interactive latency while supporting a large number of users at a high QoS. Two variations - DIVeR-2 and DIVeR-3 - are discussed.*

Numerical results utilizing both schemes are provided using data from an MPEG coded video of "Star Wars" and a model of the Seagate Barracuda disk.

1 Introduction

A Video-on-demand (VOD) service [3, 4, 7, 12, 10, 9] offers the customer the luxury of watching any of the available videos at any given time without having to leave the comforts of his home. The customer can perform a host of interactive operations - increase/decrease playback rate, jump forward/back to any location in the video sequence, pause/stop the video and subsequently resume playback.

1.1 Grouping MPEG Frames

The international MPEG coding standard [6] is the most popular coding scheme for VOD applications. As indicated in Figure 1(a), we will denote the number of P frames between two consecutive I frames by $(m-1)$, and the number of B frames between two consecutive I/P frames by $(q-1)$. Each frame is assigned a frame number which depends on its position in the playback order. The pattern indicated in Figure 1(a) repeats, i.e., frame number 30 is an I frame etc.

A video segment is a group of frames and is considered to be an atomic (indivisible) unit for retrieval purposes [8]. When a user wishes to view the video at a rate (say n_1 fps) faster than the normal playback rate (say n fps), the commonly used approach is to skip certain frames and display others at the same rate as that of normal playback. The choice of the frames to skip characterizes the effect of fast playback created. On an average, only n out of every n_1 frames has to be displayed. Skipping of frames during display implies that retrieval of some segments can be skipped. The decoding of all frames belonging to retrieved segments may not be possible due to the unavailability of frames required for the purpose. Frames that cannot be decoded have to be discarded and the number of such discarded frames has to be kept low, if not eliminated.

Simple and efficient schemes are needed for grouping MPEG frames into segments (for any desired segment length) and for determining the segments to be retrieved during fast playback (for any desired fast playback rate). In [1], a segment constitutes all frames from one I frame to (and excluding) the next. The number of frames per segment, here, is qm and fixed by the coding parameters q and m. In [8], we proposed a generalized grouping and retrieval scheme that could accommodate any desired segment length as well as fast playback rate. However, some frames belonging to retrieved segments have to be discarded in that scheme. In this paper, we propose a grouping and retrieval scheme that not only handles any desired segment length and fast playback rate but also eliminates frame discarding during fast playback.

1.2 Scheduling Retrieval of Multiple Users

The system we consider is as follows. The digitized video is stored in a disk-array based video server, transported over a broadband network (such as an ATM network) and displayed on the customer's television set via a set-top box. Prior to storage, video frames are grouped into segments according to the proposed scheme and a segment is an atomic unit of retrieval. The disk arrays are grouped together to form disk subsystems and each video is stored in its entirety in one disk subsystem. Each video segment is striped [2] across all the disks in the subsystem and the number of disks in a subsystem is determined based on the desired stripe width.

In [11], the Quasi-static Interactive Video Retrieval

0-7695-0253-9/99 $10.00 © 1999 IEEE

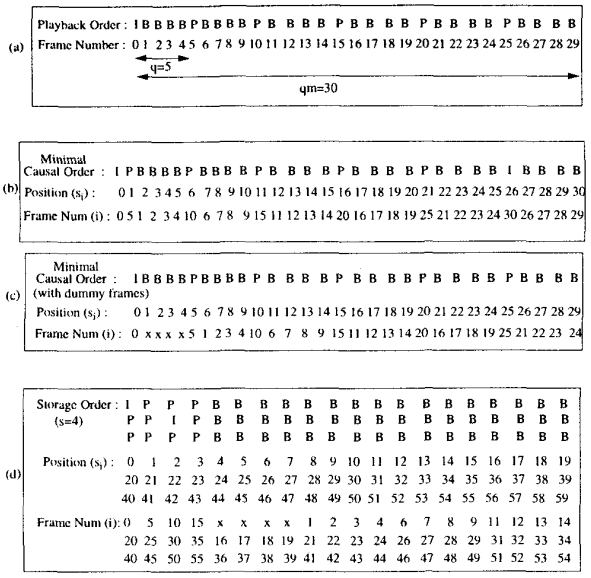

Figure 1: Frame storage order

(QuIVeR) protocol was proposed for video retrieval from disk-array based servers. QuIVeR is based on service rounds, with the number of segment retrievals for a user in a round depending on the user's playback rate (which is constant in a round). The number of groups in a round is fixed and retrieval of segments assigned to a group is done using the scan algorithm [13]. Assignment of segments to groups is such that buffer overflow and video starvation are avoided. While a large number of customers can be supported by the system, the interactive latency due to user requests may be unacceptable to the user. This is because any request by a customer is deferred until the start of the next round. The Dynamic Interactive Video Retrieval (DIVeR) protocol proposed in this paper aims at decreasing this interactive latency without any decrease in the number of supported users.

1.3 Paper Organization

The rest of the paper is organized as follows. Section 2 describes the proposed grouping scheme and segment retrieval during fast playback. DIVeR-2 and DIVeR-3 (which we shall jointly refer to as DIVeR-2,3 henceforth) are discussed in Section 3. Numerical results obtained using DIVeR-2,3 and the proposed MPEG grouping scheme are presented in Section 4. Finally, we conclude in Section 5.

2 Grouping Scheme

The primary goal of this grouping and retrieval scheme is to eliminate any discarded frames during fast playback, i.e., all frames belonging to retrieved segments are displayed.

2.1 Frame Storage Order

As seen in Figure 1(a), frames are assigned numbers according to their *playback order*. The playback order is not causal with regard to frame requirement for frame decoding. A frame in the playback order may require frames later in the order for its decoding. Since storage and retrieval of MPEG frames according to a causal order results in a simpler decoding of frames, a causal order is desirable.

We shall define the *frame storage order* as a causal arrangement of frames using which frames belonging to a video segment may be directly determined. Every set of s consecutive frames beginning at the start of the frame storage order are grouped together to form a segment.

One causal order, referred to as the *minimal causal order*, is obtained by moving each I/P frame in the playback order $(q-1)$ places ahead (see Figure 1(b)). Figure 1(c) introduces $(q-1)$ dummy B frames between the first I and P frame so that the start of the video is similar to the rest of the video. In [8], the minimal causal order (with the dummy frames) was chosen as the frame storage order and the grouping and retrieval issues were discussed.

Observation 1: When the minimal causal order is used as the frame storage order, during fast playback some frames belonging to retrieved segments may have to be dropped.

Our primary aim here is to eliminate the discarding of any retrieved frame. The frame storage order that achieves this (as will be seen later) is obtained by suitably modifying the minimal causal order (with the dummy B frames). Let s denote the segment length. For every set of s neighboring I/P frames starting at the beginning of the minimal causal order (with the dummy B frames), the last $(s-1)$ I/P frames in this set are moved ahead to immediately follow the first I/P frame in this set. The frame storage order depends on the segment length s and is shown in Figure 1(d) for the case $s=4$. The segments resulting from this frame storage order are shown in Figure 2(a).

2.2 Grouping Properties

Each segment is assigned a *segment number* with the first segment being assigned a number zero. We shall refer to the arrangement of segments in ascending order of segment numbers as the *segment storage order*. Define a *segment set* to be a set comprising the smallest number (say, c) of consecutive segments in the segment storage order that form a segment pattern. The first segment set includes c consecutive segments beginning with the first segment of the segment storage order, i.e., segment number 0.

We will use the following notation to denote different segment types :

- I_0IP segment : A segment that contains an I frame in position 0 and the remaining $(s-1)$ frames are either I or P frames.
- P_0IP segment : A segment that contains a P frame in position 0 and the remaining $(s-1)$ frames are either I or P frames with at least one I frame.
- I segment : Either an I_0IP or a P_0IP segment, i.e., a segment with at least one I frame.
- P segment : A segment that contains s P frames.
- I/P segment : Either an I or a P segment.
- B segment : A segment that contains s B frames.

Property 1: The total number of segments in a segment set equals $c = \frac{q \times LCM(m,s)}{s}$.

Property 2: (a) The segment with number iq for $i = \{0, 1, \cdots\}$ contains frames with numbers $isq, (is+1)q, (is+2)q, \cdots, (is+s-1)q$.

(b) The segment with number $(iq+j)$ for $i = \{0, 1, \cdots\}$ and $j = \{1, 2, \cdots, q-1\}$ contains the l-th B frame (in the frame storage order), where $l = i(q-1)s+(j-1)s, i(q-1)s+(j-1)s+1, i(q-1)s+(j-1)s+2, \cdots, i(q-1)s+(j-1)s+s-1$.

(c) The l-th B frame in the frame storage order has a frame number $\lfloor l/(q-1) \rfloor q + (l \ mod \ (q-1)) + 1 - q$. Here, $l = \{0, 1, \cdots\}$.

Property 2 can be used to determine the frames contained in any segment. Given any segment number, it provides closed form equations to determine the frame number of each of the s frames contained in the segment.

2.3 Fast Playback

Let n and n_1 fps be the normal and fast playback rates respectively, where $n_1 > n$. On an average, n out of every n_1 frames need to be displayed. Since all frames belonging to retrieved segments are displayed, n out of every n_1 segments needs to be retrieved. Canceling the highest common factor (HCF) between the two, $\frac{n}{HCF(n,n_1)}$ out of every $\frac{n_1}{HCF(n,n_1)}$ segments need to be retrieved. Once the choice of segments to be retrieved is determined for an integral number of segment sets, subsequent retrievals are determined in a trivial manner. Hence, we need to consider the retrieval of $LCM(c, \frac{n_1}{HCF(n,n_1)}) \times \frac{n}{n_1}$ out of $LCM(c, \frac{n_1}{HCF(n,n_1)})$ segments. We shall denote the former value by a_s and the latter by b_s, each of which is an integer. Such a retrieval results in the display of $a_f = s \times a_s$ out of every $b_f = s \times b_s$ frames.

In the rest of the discussion, we will assume that an I segment contains exactly one I frame, i.e., that $s \leq m$. The extension to the case where this assumption is relaxed is conceptually similar. Properties for this case can be similarly derived and we will not consider it here.

Property 3: Out of the c segments in a segment set, the number of I segments equals $\frac{LCM(m,s)}{m}$.

Property 4: The position of the I frame in the i-th I segment of any segment set is $mi \ mod \ s$. Here, $i = \{0, 1, \cdots, \frac{LCM(m,s)}{m} - 1\}$.

Property 5: Out of the c segments in a segment set, the number of P segments between the i-th and the $(i+1)$-th I segment equals $\lfloor \frac{m-(s-(mi \ mod \ s))}{s} \rfloor$. Here, $i = \{0, 1, \cdots, \frac{LCM(m,s)}{m} - 1\}$.

Since a segment set forms a pattern of segments, Properties 4 and 5 also hold good for the i-th I segment in the entire storage order (and not merely the segment set).

Property 6: Out of the c segments in a segment set, the number of B segments equals $\frac{q-1}{q} \times c$.

Property 7: The maximum number of P segments between any two neighboring I segments (in the segment storage order) is that between the last I segment of a segment set and the first I segment of the next segment set. This number equals $\lfloor \frac{m-1}{s} \rfloor$.

For segments arranged according to the segment storage order, the following decoding facts hold :

Decoding Facts

1. Any I_0IP segment may be decoded independently.
2. Any P_0IP or P segment only requires the previous I/P segment for the successful decoding of all its frames.
3. The first $\lceil \frac{q-1}{s} \rceil$ B segments following any I/P segment not only require this I/P segment but also the I/P segment prior to it for the successful decoding of all its frames.
4. The subsequent $(q-1-\lceil \frac{q-1}{s} \rceil)$ B segments only require this I/P segment for frame decoding.

In the above, when a certain segment is required, we mean that the decoded frames of this segment are required. Decoding the frames of the segment that is required may itself require other segments.

Based on the above facts, in order to guarantee that no retrieved frames are discarded, the following retrieval guidelines need to be adhered to :

Retrieval Guidelines

1. The I_0IP segment needs to be retrieved prior to the retrieval of any segment following it and preceding the next I_0IP segment.
2. The I/P segments following an I_0IP segment and preceding the next I_0IP segment need to be retrieved according to increasing segment numbers, i.e., an I/P segment earlier in the storage order needs to be retrieved prior to the retrieval of one appearing later in the order (for all I/P segments between two neighboring I_0IP segments.)
3. Each of the first $\lceil \frac{q-1}{s} \rceil$ B segments following an I/P segment can be retrieved after the retrieval of this I/P segment as well as the I/P segment prior to it.
4. The subsequent $(q-1-\lceil \frac{q-1}{s} \rceil)$ B segments can be retrieved after this I/P segment has been retrieved.

In order to display a_f out of b_f frames, we need to retrieve a_s out of b_s segments. Sticking to the guidelines above and based on the desired size of normal playback pieces, several segment retrieval algorithms are possible. Three possible retrievals are illustrated in Figure 2(b) for the case $n = 30$, $n_1 = 54$ fps. This fast playback rate requires $a_f = 100$ out of $b_f = 180$ frames to be displayed, or $a_s = 25$ out of $b_s = 45$ segments to be retrieved. The segments enclosed in the dashed box are the I/P segments while the others are B segments. In retrieval (iii), since segment 40 is not retrieved, following guideline (3), the first B segment following the next I/P segment (i.e., segment 46) cannot be decoded. For the same reason, segment 1 cannot be decoded and is not retrieved.

2.4 Other Interactive Operations

So far, we have been concerned with segment retrieval during fast playback in the forward direction. However, for applications such as VOD, the user is permitted a variety of other interactive operations. We shall briefly discuss the retrieval issues with the proposed grouping scheme during these interactive operations.

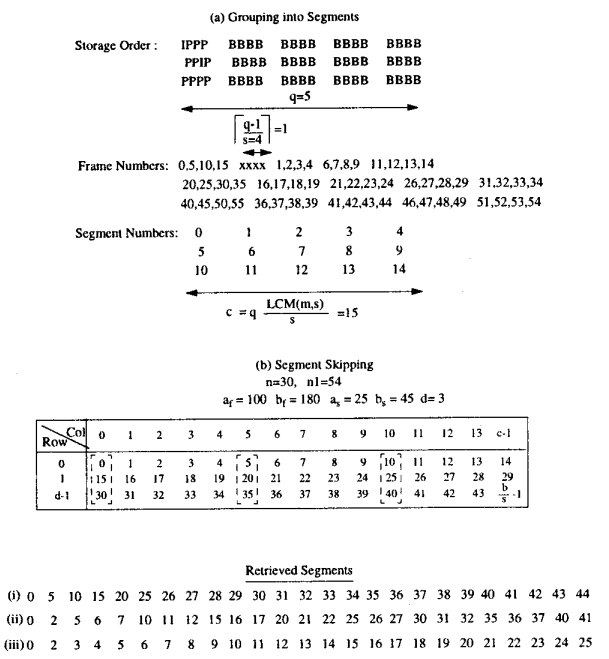

Figure 2: (a) Segment grouping (b) Segment retrieval

A slow/normal forward play retrieves segments according to the segment storage order. All segments are retrieved, i.e., no segments are skipped. A *jump* forward/backward can only be made to an I frame belonging to an I_0IP segment. If this is not the case (i.e., if the jump needs to be made to a segment other than the I segment in an I_0IP segment), some frames belonging to retrieved segments will have to be discarded.

The decoding facts and retrieval guidelines listed earlier are independent of the playback direction (forward or reverse) or speed (slow, normal or fast). When reverse playback (of any speed) is considered, the order in which segments are to be retrieved is different from that in the forward direction. We propose **Procedure R** to determine the order of segment retrieval for slow/normal reverse playback, i.e., when all segments are retrieved.

Procedure R

Input:
x_1 = segment set from which reverse playback begins
$\frac{LCM(m,s)}{m}$ = number of I segments in a segment set

Output:
Segments in the order they should be retrieved

begin
begin declaration
i = segment set from which retrieval is currently taking place
j = I segment to be retrieved from segment set i
end declaration

for $i = x_1$ to 0 **do**
for $j = \frac{LCM(m,s)}{m} - 1$ to 0 **do**
Retrieve j-th I segment in segment set i
Retrieve P segments following this I segment (and before the next one) in increasing segment numbers
Retrieve B segments following the retrieved I/P segments in decreasing segment numbers
end for
end for
end

Theorem 1: For slow/normal reverse playback using the proposed grouping scheme, **Procedure R** results in the least number of prefetched segments prior to start of playback (in order to avoid video starvation). This number equals $1 + \lfloor \frac{m-1}{s} \rfloor$.

As in the case of fast forward, fast reverse needs to skip retrieval of certain segments. The choice of segments to skip again depends on the same two issues that were considered in the fast forward case - desired effect of fast display (i.e., size of pieces) and the set of retrieval guidelines.

3 DIVeR-2 and DIVeR-3

The framework around which the DIVeR-2 and DIVeR-3 protocols are built is as follows :

- Segment skipping is employed during fast playback utilizing the proposed scheme for choosing segments to be retrieved during fast playback.
- Of all playback rates that the system offers a user, the fast rates (i.e., rates faster than the normal rate r_n) can be chosen arbitrarily while each slow rate (i.e., slower than the normal rate r_n) should be equal to r_n/m for some positive integer m.
- Segment retrievals are grouped together and the scan algorithm is used within a group. Each group has the same duration T_g which equals the time that a segment lasts when played back at the normal rate.

The above ensures that $f(r)$, the duration that a segment lasts when played back at any valid rate r, is an integral multiple of the group duration T_g.

Let k be the number of segments that are prefetched prior to the start of playback. The value of k depends on the particular scheme used for grouping frames into segments [8]. We wish to restrict the buffer occupancy at each user's set-top box to a value between $k-1$ and $k+1$ segments. If the buffer occupancy falls below $k-1$ segments, there exists a possibility of video starvation. The frame that needs to be displayed may belong to an unretrieved segment, or another frame necessary for decoding the frame to be displayed may not be retrieved. If the buffer occupancy is greater than $k+1$ segments, buffer overflow occurs because the buffer capacity is set at $k+1$ segments. In order to maintain the buffer occupancy between $k-1$ and $k+1$ segments, one

video segment has to be retrieved in the duration that a previously retrieved segment is played back.

When the scan algorithm is used for segment retrieval in each group, the segments to be retrieved in a particular group need to be known prior to start of retrieval in that group. Moreover, a segment can be retrieved anywhere within the group. Hence, in order to guarantee retrieval of one segment during playback of another, the following *Requirement Set R* needs to be satisfied :

- At least one group should be encompassed by the playback of each video segment. Portions of the same video segment may be played back at different rates when a user requests a rate change in the middle of a segment playback.
- The segment retrieval needs to be assigned to one of these groups (encompassed by the playback of a video segment) prior to the start of retrieval in that group.

When the user makes a request for a change of playback rate, it is desirable that the playback rate is changed to the new rate with minimum latency. However, in order to guarantee avoidance of video starvation and buffer overflow, the protocol should also ensure that requirement set R is satisfied. DIVeR-2 and DIVeR-3 ensure this.

3.1 DIVeR-2

The choice of allowable playback rates and group duration dictated by DIVeR-2,3 results in the following desirable property :

- The duration that a segment lasts when played back at any allowable rate (without any rate change) is an integral multiple of the group duration.

If a video segment playback begins at the start of a group, then the playback of that video segment is completed at the end of some group. The particular group at which playback ends depends on the user's playback rate. Playback of the next video segment for the same user begins at the start of this group and the process continues. Since at least one group is encompassed by the playback of a video segment, retrieval of one video segment takes place in one of these groups that is encompassed by the playback of a video segment. In other words, requirement set R is satisfied.

In order to utilize the property discussed above, the following is employed in DIVeR-2:

- User interactive requests are deferred until the playback of the user's currently played back video segment is completed.

Of all groups that are encompassed by the playback of a user's video segment, one needs to be selected for a segment retrieval for that user. Different criteria may be used for selecting the group in which the segment retrieval takes place [11]. A good criterion should not only result in a small variance in the number of segment retrievals assigned to groups but should also be simple to implement. A small variance in the number of segment retrievals between groups results in a decrease in the number of segments that cannot be retrieved due to a lack of time. DIVeR-2 uses the following criterion :

- Of all groups that are encompassed by the playback of a user's video segment, the first one is chosen for retrieving a segment for that user.

In the section on numerical results, it is seen that such an assignment of segments to groups results in a small variance in the number of assignments to groups.

When a user requests start of playback or a jump to a particular location in the video sequence, k video segments need to be prefetched which are assigned to be retrieved in consecutive groups. In addition, a request made after retrieval in a group has begun has to be deferred until the start of the next group. Hence, these two operations have a latency of $U[0, T_g) + kT_g$. For the start operation, no frames are displayed during the latency period. For the jump operation, frames belonging to buffered segments may be displayed during the latency period if they can be decoded failing which the last displayed frame is frozen on the screen. When a user requests a pause/stop, resume or change of playback rate, the interactive latency is till the playback of the currently displayed segment is completed. These results are tabulated in Table 1 where r_c denotes the user's playback rate at the time the request is made and U denotes the uniform distribution.

3.2 DIVeR-3

In order to further decrease the interactive latency due to user requests, the following is done in DIVeR-3 :

- User interactive requests are deferred just until that time when serving the request results in completion of playback of the currently displayed segment at a group boundary.

When this is done, playback of a segment (at one or more rates) starts at a group boundary and ends at another. Hence, requirement set R is satisfied and consequently, it is possible to retrieve a segment during the display of another.

The choice of groups for segment retrieval in DIVeR-3 is similar to DIVeR-2 :

- Of all groups that are encompassed by the playback of a user's video segment, the first one is chosen for retrieving a segment for that user.

The numerical results again illustrate that such an assignment of segments to groups results in a small variance between assignment to groups.

The interactive latency of DIVeR-3 is less than that of DIVeR-2 and is tabulated in Table 2.

4 Numerical Results

Let the disk system contain 1000 videos each lasting 100 minutes when played back at 30 fps. The total storage required, assuming MPEG-2 compressed video, is 3 T Bytes which requires $\lceil \frac{3000}{9.1} \rceil = 330$ disks assuming a disk storage capacity of 9.1 GBytes. If the number of disks per subsystem is 10, the number of disk subsystems is 33. So, each disk subsystem contains either 30 or 31 videos.

Numerical results are presented using the MPEG-1 coded data of the *Star Wars* video [5]. Grouping of frames into video segments and the subsequent retrieval of segments during fast playback are based on the proposed scheme. The disk drive is taken to be the Seagate ST19171FC

Interactive operation	Latency	Display during latency
Start playback	$U[0, T_g) + kT_g$	none
Jump ahead/back	$U[0, T_g) + kT_g$	frame prior to request/frozen
Pause/stop	$U[0, f(r_c))$	frame prior to request
Resume	$U[0, f(r_c))$	frame frozen during pause
Increase/decrease rate	$U[0, f(r_c))$	frame prior to request

Table 1: Latency of user requests in DIVeR-2

Interactive operation	Latency	Display during latency
Start playback	$U[0, T_g) + kT_g$	none
Jump ahead/back	$U[0, T_g) + kT_g$	frame prior to request/frozen
Pause/stop	$U[0, T_g)$	frame prior to request
Resume	$U[0, T_g)$	frame frozen during pause
Increase/decrease rate	$U[0, T_g)$	frame prior to request

Table 2: Latency of user requests in DIVeR-3

Barracuda-9 3.5 inch hard drive and is modelled as described in [11].

Let the set of playback rates available to each user be : 3.75, 7.5, 15, 30, 60, 120, 240 fps. Each of these playback rates corresponds to a state the user is in. Consecutive state numbers are assigned to increasing rates with the slowest rate (3.75 fps) being assigned to state 0. The behavior of each user is assumed to be independent and identically distributed (i.i.d.). The transition probability matrix $P = [p_{ij}]_{7\times7}$ for the user playback rates is :

$$P = \begin{pmatrix} 0.00 & 0.20 & 0.20 & 0.50 & 0.06 & 0.03 & 0.01 \\ 0.20 & 0.00 & 0.20 & 0.50 & 0.05 & 0.03 & 0.02 \\ 0.20 & 0.20 & 0.00 & 0.50 & 0.05 & 0.03 & 0.02 \\ 0.15 & 0.15 & 0.20 & 0.00 & 0.20 & 0.15 & 0.15 \\ 0.02 & 0.03 & 0.05 & 0.50 & 0.00 & 0.20 & 0.20 \\ 0.02 & 0.03 & 0.05 & 0.50 & 0.20 & 0.00 & 0.20 \\ 0.01 & 0.03 & 0.06 & 0.50 & 0.20 & 0.20 & 0.00 \end{pmatrix}$$

where p_{ij} denotes the conditional probability that the next state is j given that the current state is i and that the user has just made a request. The average time (in sec) spent by a user in a particular state i is exponentially distributed with mean T_i, for $i = \{1, 2, \cdots, 7\}$ and :

$[T_1\ T_2 \cdots\ T_7] = [5\ 5\ 5\ 600\ 5\ 5\ 5]$

With 12 frames/segment and 30 fps as the normal playback rate, the duration of a group is $T_g = \frac{12\times1000}{30} = 400$ ms.

Table 3 illustrates the performance of DIVeR-2 and DIVeR-3 protocols by tabulating the fraction of unretrieved segments and the disk utilization. The performance of QuIVeR-5 is also included for comparison [11]. It is observed that the performance of the three protocols concerning these two parameters is not significantly different. However, DIVeR-2 is superior to QuIVeR-5 regarding interactive latency, while DIVeR-3 has the lowest interactive latency of them all.

The variance was observed to be quite small. For DIVeR-2, it was in the range [0.30,0.44]; while for DIVeR-3, it was in the range [0.28,0.41].

For the set of playback rates being considered and with 12 frames/segment, we have :

T_g = group time = 400 ms

T_c = time to play back a segment at the slowest rate (3.75 fps) = $\frac{12\times1000}{3.75}$ = 3200 ms

The interactive latency for each protocol is as indicated in Table 4. The start and jump operations assume that playback after the operations takes place at the normal rate. When the pause/stop, resume or rate change request is made in DIVeR-2, the latency depends on the user's current playback rate. The values 400/800/1600/3200 in the uniform distribution are for the rates 30(or higher)/15/7.5/3.75 fps respectively. Thus, we see that the latency improves as we go from QuIVeR-5 to DIVeR-2 to DIVeR-3.

5 Conclusions

In this paper, we proposed a scheme for grouping MPEG frames into segments of any desired length. Closed form equations were provided that can facilitate the determination of which frames need to be grouped together to form a segment. Guidelines were provided for handling a variety of interactive operations - slow/normal/fast forward/reverse play and jump forward/backward. During fast forward/reverse playback, retrieval of some segments needs to be skipped. Using these guidelines and depending on the desired effect of fast playback, suitable retrieval algorithms may be easily devised. The grouping and retrieval scheme is such that no retrieved frames are ever discarded.

We then proposed the DIVeR protocol for scheduling segment retrieval from disk array based video servers when several users are using the system simultaneously. DIVeR is an

# Usr	Fraction of unretrieved segments			Disk utilization		
	Q5	D2	D3	Q5	D2	D3
50	.000000	.000000	.000000	.79438	.794203	.793495
51	.000000	.000000	.000000	.80893	.807920	.807228
52	.000000	.000000	.000000	.82157	.820274	.822146
53	.000002	.000005	.000000	.83545	.833373	.834440
54	.000012	.000007	.000007	.84887	.847578	.848469
55	.000046	.000021	.000064	.86199	.861496	.862003
56	.000090	.000070	.000117	.87634	.873864	.876151
57	.000249	.000205	.000225	.88980	.888560	.889085
58	.000540	.000566	.000614	.90406	.903190	.902846
59	.001257	.001292	.001245	.91683	.916416	.916974
60	.002153	.002229	.002373	.92884	.929208	.928944

Table 3: Protocol comparison

User Request	Interactive latency		
	Q5	D2	D3
Start, Jump	$U[0, 3200) + 400k$	$U[0, 400) + 400k$	$U[0, 400) + 400k$
Pause/stop, Resume Rate change	$U[0, 3200)$	$U[0, 400/800/1600/3200)$	$U[0, 400)$

Table 4: Interactive latency

extension to the QuIVeR protocol and is aimed at decreasing the interactive latency due to user requests. DIVeR-2 has a lower interactive latency than the QuIVeR protocols, while DIVeR-3 has an interactive latency lower than that of DIVeR-2. Numerical results show that the performance of the two DIVeR protocols is comparable to the most superior QuIVeR protocol (QuIVeR-5) with regard to the fraction of unretrieved segments and the disk utilization. DIVeR-2 and DIVeR-3 are simple to implement and are well suited for a real time application such as VOD.

References

[1] Ming-Syan Chen, Dilip D. Kandlur, and Philip S. Yu. "Storage and Retrieval Methods to support fully interactive playout in a disk-array-based video server". *Multimedia Systems*, 3:126–135, 1995.

[2] P. M. Chen and E. K. Lee. "Striping in a RAID Level 5 Disk Array". *Proceedings Joint International Conference on Measurement & Modelling of Computer Systems, Sigmetrics '95/Performance '95*, pages 136–145, May 1995.

[3] Daniel Deloddere, Willem Verbiest, and Henri Verhille. "Interactive Video On Demand". *IEEE Communications Magazine*, pages 82–88, May 1994.

[4] Yurdaer N. Doganata and Asser N. Tantawi. "Making a cost-effective Video Server". *IEEE Multimedia*, 1994.

[5] Mark W. Garrett and Walter Willinger. "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic". *SIGCOMM*, pages 269–280, 1994.

[6] Pramod Pancha and Magda El Zarki. "MPEG Coding for Variable Bit Rate Video Transmission". *IEEE Communications Magazine*, pages 54–66, May 1994.

[7] P. Venkat Rangan, Harrick M. Vin, and Srinivas Ramanathan. "Designing an On-Demand Multimedia Service". *IEEE Communications Magazine*, pages 56–64, July 1992.

[8] Senthil Sengodan and Victor O.K. Li. "A Generalized Grouping and Retrieval Scheme for Stored MPEG Video". *IEEE ICC*, June 1997.

[9] Senthil Sengodan and Victor O.K. Li. "A Quasi-static Retrieval Scheme for Interactive VOD Servers". *Computer Communications*, 1997.

[10] Senthil Sengodan and Victor O.K. Li. "A Shared Buffer Architecture for Interactive VOD Servers". *IEEE INFOCOM*, April 1997.

[11] Senthil Sengodan and Victor O.K. Li. "QuIVeR : A Class of Interactive Video Retrieval Protocols". *IEEE ICMCS*, June 1997.

[12] W. D. Sincoskie. "System Architecture for a Large Scale Video-on-Demand Service". *Computer Networks and ISDN Systems*, 22:155–162, 1991.

[13] T.L.Kunii, Y.Shinagawa, R.M.Paul, M.F.Khan, and A.A.Khokar. "Issues in Storage and Retrieval of Multimedia Data". *Multimedia Systems*, 3:298–304, 1995.