The HKU Scholars Hub The University of Hong Kong 香港大學學術庫

Title	QuIVeR: A class of interactive video retrieval protocols
Author(s)	Sengodan, Senthil; Li, Victor OK
Citation	International Conference On Multimedia Computing And Systems -Proceedings, 1997, p. 186-193
Issued Date	1997
URL	http://hdl.handle.net/10722/46016
Rights	Creative Commons: Attribution 3.0 Hong Kong License

QuIVeR : A Class of Interactive Video Retrieval Protocols*

Senthil Sengodan, Victor O. K. Li

Communication Sciences Institute
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2565, U.S.A.
E-mail: {sengodan,vli}@milly.usc.edu

Abstract

Video-on-demand (VOD) servers need to be efficiently designed in order to support a large number of users viewing the same or different videos at different rates. In this paper, we propose the Quasi-static Interactive Video Retrieval (QuIVeR) Protocol for this purpose when disk-array based video servers are used. Five variations - QuIVeR-1, QuIVeR-2, QuIVeR-3, QuIVeR-4 and QuIVeR-5 - are presented. The properties as well as the relative merits and demerits of each protocol are discussed. The protocols require no buffer at the server and hence, all retrieved segments are immediately transmitted to the appropriate users. The amount of buffer required at each user's set-top box is reduced to two video segments. Guarantees are provided for the avoidance of video starvation as well as buffer overflow at each user's set-top box. Numerical results, obtained using data from an MPEG coded "Star Wars" video, are provided.

1 Introduction

In a Video-on-Demand (VOD) system [3, 4, 7, 8, 14], a user may view any of the available videos at any desired time. The user may perform several interactive operations such as pause/stop, slow/fast forward and jump forward/backward. We believe that reverse play is not necessary in such a system when jump backward is provided. Moreover, providing reverse play when the video coding uses inter-frame compression (as in MPEG) usually requires a larger buffer. The VOD system offers the customer a wide range of discrete fast/slow viewing speeds. A good VOD retrieval scheme should handle startup and interactive requests with a low latency, require small buffers at the user's set-top box and support a large number of users with a guaranteed Quality of Service (QoS).

Segment retrievals occur in service rounds which are durations of time within which all users are served their required number of video segments. Several disk scheduling schemes which are based on service rounds have been proposed in the literature. These include round-robin scheduling [6, 9], scan scheduling [6, 9] and grouped sweeping scheduling (GSS) [1]. When no buffer is provided at the server and all segments retrieved from disk are immediately transmitted to the user, the buffer requirement at each user's set-top box while any of these scheduling schemes are used depends on the user playback rate. In the absence of smoothing buffers at the server, many studies [1] indicate that the total buffer requirement is the sum of the individual users' buffer (and any staging buffer) requirement which is taken to vary with playback rates. However, any buffer provided at a user's set-top box is fixed and does not depend on the user's playback rate. Consequently, the total buffer requirement needs to be minimized taking into account the fact that the buffer at a user's set-top box is a constant. The proposed protocol, QuIVeR, achieves this.

The rest of the paper is organized as follows. Section 2 describes aspects dealing with video storage and the user model. Section 3 describes general features of the proposed QuIVeR protocol. Section 4 describes each of the five versions of QuIVeR, while Section 5 illustrates the protocol performance by numerical examples. Finally, Section 6 concludes the paper.

2 Video storage, user model

2.1 Video storage

The online storage medium is taken to be a disk system due to its relatively low cost (compared to RAM) and small retrieval latency (compared to tape). The surface of a disk contains concentric circles called tracks on which data is stored [10]. In order to retrieve any data, the disk head needs to be moved from the track that it is currently positioned over to the track containing the data. This operation is referred to as a disk seek. After a disk seek has been performed, the head needs to wait until the data is positioned below the head by the rotating disk. This time is referred to as the rotational latency. Finally, the data is actually retrieved in a time referred to as the data transfer time. The total time for retrieval of data is, thus, the sum of the disk seek, rotational latency and data transfer time.

Frames are grouped into segments, where each segment typically consists of the same number of frames [2, 11, 13]. A segment is the smallest information unit that may be retrieved by the disk subsystem for a user at any time. The larger the segment size, the larger is the startup (as well as interactive) latency (due to an increase in the duration of the service round), the larger is the buffer requirement at each user's set-top box and the larger is the number of users that may be accommodated in the system (since a disk seek is now for a larger amount of retrieved data.) Our aim in this paper is to minimize the buffer requirement and provide guarantees for avoiding video starvation as well as buffer overflow for a given segment size.

The disk storage system comprises several disk subsystems, each comprising the same number of disks. A video

*This research is supported in part by the TRW Foundation and in part by the Pacific Bell External Technology Program.

0-8186-7819-4/97 $10.00 © 1997 IEEE

is stored in its entirety in one disk subsystem and each disk subsystem comprises several videos. A video segment is uniformly striped across all the disks in the disk subsystem. If S_s is the size of a segment and N_d is the number of disks in the subsystem, then the amount of data retrieved by one disk for any user at any particular time equals $\frac{S_s}{N_d}$. This data should be stored in such a manner that the retrieval time is bounded by a desirably low value. Techniques for achieving this have been described in [15].

2.2 User model

Every user can view each video at N_r different playback rates. These rates are denoted as r_i frames per second (fps) where $i = \{0, 1, \cdots, N_r - 1\}$, and without loss of generality $r_i < r_j$, $\forall i < j$. The normal playback rate (usually 30 fps) is denoted as r_n. While fast playback rates ($> r_n$) can assume any desired value, QuIVeR has certain restrictions on slow playback rates ($< r_n$), the reasoning behind which is explained later. In QuIVeR-1,2,3,4, each slow playback rate has to be an integer multiple of the slowest rate r_0. In QuIVeR-5, in addition to each slow playback rate being an integer multiple of r_0, it also has to be a factor of r_n. Each user can pause/stop the video at any desired time. A user can also jump forward/backward in the video sequence. Since such a jump operation does not result in a change in the number of retrieved segments per round, it is not included in the user model.

The user state transition chain depicting the user model comprises $m = N_r + 1$ states since we have one stop/pause state and N_r different playback rates. We assume that the behaviour of each user is independent and identically distributed (i.i.d.). The transition probability matrix for the user states is $P = [p_{ij}]_{m \times m}$ 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. A request is either a rate change or a pause/stop of video.

The amount of time spent by a user in a particular state is assumed to be exponentially distributed with mean $T_i = \frac{1}{\mu_i}$. Transforming the user state transition chain into a markov chain with m states, we have $\forall i, j = 0, 1, \cdots, m-1$:

μ_{ij} = transition rate from state i to state $j = \mu_i p_{ij}$

We can then solve the balance equations to determine the steady state probability π_i that the user is in state i.

3 QuIVeR operation

The following assumptions are made for the rest of the discussion :

1. Segment skipping is employed during fast playback.
2. Suitable schemes exist for choosing segments to be retrieved during fast playback.

Assumption 1 ensures that a fast playback at any rate requires the same number of segment retrievals in a service round as that required by the normal rate. Some schemes for grouping frames into segments and choosing segments during fast playback can be found in [11, 13].

QuIVeR has the following requirements/restrictions:

- T_c = service round duration = time that a video segment lasts when played back at the slowest rate, r_0.

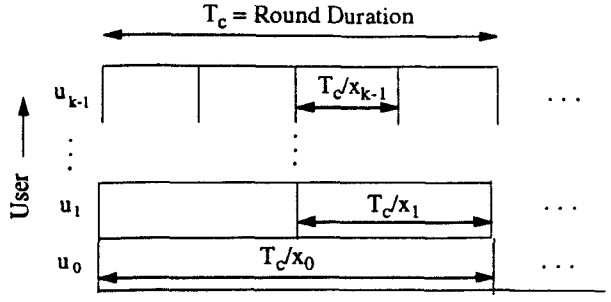

Figure 1: Illustration of macro slots

- Let the set of allowable playback rates be $\{r_i\}$, where $i = \{0, 1, \cdots, N_r - 1\}$ and without loss of generality $r_a < r_b$, $\forall a < b$. Let the slowest rate be r_0, the normal rate be r_n and $g = r_n/r_0$. The fast rates $r_j > r_n$ can be chosen arbitrarily, but all slow rates $r_j \leq r_n$ should be an integral multiple of r_0.

A *macro-slot* of a user is a duration of time within which one segment is retrieved while another is played back for that user. Each user i at a playback rate r_{ui} has $f(r_{ui})$ macro-slots in a service round, each of duration $\frac{T_c}{f(r_{ui})}$ where

$$f(r_{ui}) = \begin{cases} \frac{r_n}{r_0} & \text{if } r_{ui} \geq r_n \\ \frac{r_{ui}}{r_0} & \text{otherwise} \end{cases}$$

This is illustrated in Figure 1 where u_i indicates a user. The restrictions imposed by QuIVeR ensure that the end of the macro-slot of the last segment retrieval of any user coincides with the end of the service round.

Let k be the number of segments that are prefetched prior to start of playback. The value of k depends on the particular scheme used for grouping frames into segments [11, 13]. 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.

3.1 Startup and interactive latency

Since QuIVeR is a quasi-static protocol, all schedules for segment retrieval are determined at the start of each round. Startup and interactive (rate change, pause/stop, resume, jump) requests made after the start of a round are deferred until the start of the next round. Beginning with the next round, macro-slot durations of the user requesting startup/interaction are suitably changed.

A startup or jump request requires k segments to be prefetched only after which playback begins. This requires k macro-slots starting from the round after the one in which

the request was made. The duration of each macro-slot is $T_c \times (r_0/r_n)$ assuming that the playback starts at the normal rate (after a startup or jump request). Since the request is equally likely to be made anywhere within the service round, the time duration between the instant of request and start of the next round is $U[0, T_c)$. Here, $U[a, b)$ is a random variable uniformly distributed in the interval $[a, b)$. Hence, the latency for a start and jump request equals $U[0, T_c) + kT_c \times (r_0/r_n)$. During the latency period, there is no display on the screen.

Due to the continuity of video sequence for the remaining interactive requests (rate change, pause/stop, resume), requests made during a round take effect at the start of the next round. The interactive latency for such operations equals the time duration between the instant of request and the start of the next round which is $U[0, T_c)$. Table 1 tabulates the latency due to various user requests.

3.2 Groups

In QuIVeR, groups are created by adjacent macro-slot boundaries (see Figure 7). Any segment whose macro-slot encompasses a group can be assigned to that group. Retrieval of segments within each group takes place using the SCAN algorithm.

VOD is an application where users, for a large part, view a video at the normal playback rate. With segment skipping, the load on the disk system that a user imparts while at a fast playback rate is identical to that imparted while at the normal rate. Hence, the fraction of time that a user imparts peak load (normal playback rate or higher) on the disk system is close to unity. Consequently, the system needs to be designed for peak load. If k users are present in the system, the number of retrievals in a round that the system needs to be designed for equals $k \times \frac{r_n}{r_0}$. Here, $\frac{r_n}{r_0}$ is the number of segments retrieved in a round for a user at normal rate or higher.

Of the $k \times \frac{r_n}{r_0}$ possible segment retrievals, the maximum number assigned to a group is proportional to the group duration. Assignment of segments to groups varies between the five versions of QuIVeR and is discussed below.

4 QuIVeR versions

4.1 QuIVeR-1

The algorithm used by QuIVeR-1 to assign segment retrievals to groups is given in Figure 2. The algorithm is a Non-preemptive Earliest Deadline First (N-EDF) algorithm. Assignment of segments to groups should be such that each segment is retrieved within its macro-slot. It can be proved that the N-EDF algorithm employed by QuIVeR-1 can always achieve this [12].

4.2 QuIVeR-2

The N-EDF algorithm employed by QuIVeR-1 results in an assignment of segments to the earliest possible group [12]. Since the average seek time for a larger group is smaller than that for a smaller one, it is desirable to move retrievals assigned by N-EDF from smaller groups to larger ones when both groups are within the same macro-slot of the user. QuIVeR-2 does this resulting in better performance compared to QuIVeR-1. Figure 3 describes the algorithm used by QuIVeR-2 to move assignments from smaller groups to larger ones. This procedure is employed after *Procedure Q1-A* has been employed.

```
Procedure Q1-A

begin
    begin declaration
        g = number of groups
        max_num(i) = max number of segments allowed in group i
        S(i) = set of segments that can be assigned to group i
        U(i) = set of unassigned segments that can be assigned to group i
        num(U(i)) = number of segments in set U(i)
    end declaration
    begin initialization
        for i = group 0 to group (g-1)
            Determine S(i)
            Set U(i) = S(i)
        end for
    end initialization
    for i = group 0 to group (g-1)
        Assign min(max_num(i), num(U(i))) segments with
                    earliest deadline in U(i) to group i
        for j=group (i+1) to (g-1)
            Delete these assigned segments from U(j)
        end for
    end for
end
```

Figure 2: Procedure Q1-A

4.3 QuIVeR-3

The purpose of moving segments from smaller groups to larger ones in QuIVeR-2 was to decrease the number of retrievals in smaller groups and increase those in larger ones, thereby improving performance. QuIVeR-3 aims to achieve the same purpose by different means. The N-EDF algorithm is not employed here. Of all possible groups that a segment of a user can be assigned to, the group that can accomodate the largest number of retrievals (and which has not reached its retrieval capacity) is chosen. Segments of the same user are assigned consecutively and users are handled in decreasing order of retrievals in a round. Figure 4 describes the algorithm used by QuIVeR-3 to assign segments to groups.

4.4 QuIVeR-4

In QuIVeR-1,2,3, the number of groups in a service round (and their duration) was fixed. In an application like VOD, users predominantly are viewing videos at the normal playback rate. Hence, it is likely that in any particular service round there are no users viewing a video at one or more slow playback rates. Depending on the set of playback rates that are being used in a service round, it is advantageous to change the number of groups (and their duration) dynamically. This is done in QuIVeR-4. With the exception that the number of groups in a round is chosen dynamically, QuIVeR-4 is identical to QuIVeR-3 (see *Procedure Q3-A*).

4.5 QuIVeR-5

In QuIVeR-1,2,3, the choice of playback rates resulted in an uneven distribution of group durations. In QuIVeR-2,3, steps were taken to decrease the number of segments assigned to groups of smaller duration. QuIVeR-4 improved performance by dynamically determining the number of

Interactive operation	Latency	Display during latency
Start playback	$U[0, T_c) + kT_c \times (r_0/r_n)$	none
Jump ahead/back	$U[0, T_c) + kT_c \times (r_0/r_n)$	frame prior to request/none
Pause/stop	$U[0, T_c)$	frame prior to request
Resume	$U[0, T_c)$	frame frozen during pause
Increase/decrease rate	$U[0, T_c)$	frame prior to request

Table 1: Latency of user requests

```
Procedure Q2-A

begin
    begin declaration
        g = number of groups
        duration(i) = time duration of group i
        num(i) = number of segments in group i
        max_num(i) = max number of segments allowed in group i
        f(i) = farthest group that segment i can be assigned to
    end declaration
    for i = group 0 to group (g-1)
        for j = segment in group i in increasing deadlines (i.e., increasing f(j))
            for k=group (i+1) to group f(j)
                if ((duration(k) > duration (i)) and (num(k) < max_num(k)))
                    move segment j from group i to group m
                    set k=f(j)+1
                end if
            end for
        end for
    end for
end
```

Figure 3: Procedure Q2-A

```
Procedure Q3-A

Input : Set of users and their playback rates
Output : Set of selected groups for each user's segment

begin
    begin declaration
        g = number of groups
        dur(i) = duration of group i
        num(i) = number of segments assigned to group i
        max_num(i) = max number of segments allowed in group i
        ret(i) = number of segment retrievals for user i
        G(i,j) = set of groups within macro-slot of user i's segment j
    end declaration
    for i = each user in decreasing order of ret(i)
        for j = each segment of user i
            if each group in G(i,j) is full
                Move a user k's segment l assigned to a group m in G(i,j) to
                                        the largest non-full group in G(k,l)
            else
                m = largest non-full group in G(i,j)
            end if
            Assign user i's segment j to group m
            Increment num(m)
        end for
    end for
end
```

Figure 4: Procedure Q3-A

groups in a round depending on the user playback rates. In QuIVeR-5, the playback rates are chosen such that the group durations are identical. This also results in a simplification of segment assignment to groups.

In addition to the general requirements of QuIVeR, QuIVeR-5 has the following restrictions on slow playback rates :

- Let the set of allowable playback rates be $\{r_i\}$, where $i = \{0, 1, \cdots, N_r - 1\}$ and without loss of generality $r_a < r_b$, $\forall a < b$. Let the slowest rate be r_0, the normal rate be r_n and $g = r_n/r_0$. All slow rates $r_j < r_n$ should satisfy :
  - r_j is an integral multiple of r_0
  - r_n is an integral multiple of r_j

The above requirement leads to the following lemma.

Lemma 1: The macro-slot duration of a user at any playback rate is an integral multiple of that of a user at normal rate.

Proof: Macro-slot duration of a user at normal rate $= T_c/g$
Macro-slot duration of a user at rate r_i

$$= \begin{cases} T_c/g & \text{if } r_i \geq r_n \\ (r_n/r_i) \times (T_c/g) & \text{otherwise} \end{cases}$$

Since r_n/r_i is an integer $\forall\, r_i < r_n$, we have the result. □

The number of resulting groups is seen to be g and the time duration of each group is T_c/g.

We shall now describe the algorithms used by QuIVeR-5 to obtain a uniform distribution of segments to groups. Two algorithms - *Procedure Q5-A* and *Procedure Q5-B* - are presented and their properties discussed. The following notation is used. The number of users at rate r_i is n_{r_i}, and m_{r_i} is the number of segments to be retrieved for each user at this rate. These m_{r_i} segments will be numbered $s_0, s_1, \ldots, s_{m_{r_i}-1}$ and will be retrieved in that order. The groups will be denoted as G_i where $i = \{0, 1, \ldots, g-1\}$.

Procedure Q5-A and *Procedure Q5-B* are greedy algorithms and are described in Figures 5 and 6, respectively. Of all possible groups that a segment can be assigned to, the group with the least number of segments is chosen. In the case where more than one group with the same least number of segments exists, any of these least occupied groups can be selected. In our case, we will choose the group with the smallest number in case of a tie. The difference between the two procedures is the order in which users are presented to the procedure for the assignment of their segments to

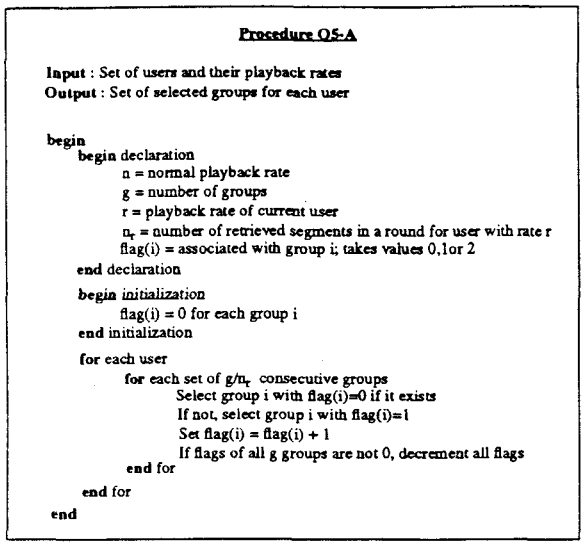

Figure 5: Procedure Q5-A

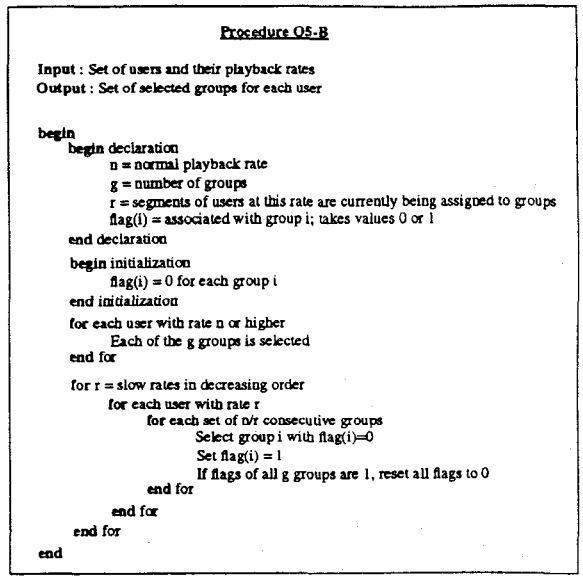

Figure 6: Procedure Q5-B

groups. *Procedure Q5-A* serves users in any random order (independent of their rates), say according to user numbers. *Procedure Q5-B* serves users according to the number of segments that need to be retrieved for the user. A user that requires a certain number of segments in a round is only served after all users that require more number of segments are served. Some properties of these algorithms are now presented. A property labeled *An* applies to *Procedure Q5-A*, *Bn* applies to *Procedure Q5-B*, and *ABn* applies to both procedures.

Property AB1: The number of segments assigned to groups encompassed by the first macro-slot of a user at any rate is never less than that assigned to groups encompassed by any other macro-slot of that user.

Proof: Let the user be denoted by i. Let his playback rate be r and the number of segments retrieved per round for user i be n_r. If the total number of groups is g, the number of groups encompassed by a macro-slot of user i is g/n_r. For any user j at any playback rate r', *Procedure Q5-A* and *Procedure Q5-B* prefer groups that appear earlier in user j's macro-slot than those that appear later. A group appearing later in user j's macro-slot is not chosen if one appearing earlier has a lesser number of segments assigned to it. The groups encompassed by a macro-slot of user i does not include more groups appearing earlier in user j's macro-slot than the groups encompassed by the first macro-slot of user i. This is true irrespective of the individual playback rates r and r'. Hence, the result. □

Property AB2: The number of segments assigned to groups encompassed by the last macro-slot of a user at any rate never exceeds that assigned to groups encompassed by any other macro-slot of that user.

Proof: The argument is similar to that in the proof of Property AB1. The groups encompassed by a macro-slot of user i includes more groups appearing earlier in user j's macro-slot than the groups encompassed by the last macro-slot of user i. This is true irrespective of the individual playback rates r and r'. Hence, the result. □

Property AB3: The number of segments assigned to any group does not exceed that assigned to G_0.

Proof: This is a direct consequence of Property AB1 when users at the normal rate or higher are considered. □

Property AB4: The number of segments assigned to any group at least equals that assigned to the last group G_{g-1}.

Proof: This is a direct consequence of Property AB2 when users at the normal rate or higher are considered. □

Property A1: If n_{G_i} is the number of segments to be retrieved in group G_i, then

$$\max_{\forall i,j} \left| n_{G_i} - n_{G_j} \right| = n_{G_0} - n_{G_{g-1}} \leq 2$$

Property B1: If n_{G_i} is the number of segments to be retrieved in group G_i, then

$$\max_{\forall i,j} \left| n_{G_i} - n_{G_j} \right| = n_{G_0} - n_{G_{g-1}} \leq 1$$

The equality in Properties A1 and B1 follows directly from Properties AB3 and AB4. The inequality in the two cases were obtained by considering the various scenarios.

5 Numerical results

Results are presented using MPEG-1 coded data of the "Star Wars" video [5]. The ratio of the number of I, P and B frames is $1:3:8$. Grouping of frames into segments and selection of segments for retrieval during fast playback is based on [11]. While such a scheme results in a larger variance of segment size, it has the advantage that no frames belonging to retrieved segments need to be discarded. The use of the SCAN algorithm within a group and the fact that the data transfer time is smaller than the seek and rotational latency implies that the variance of segment size is not of importance. Though the results have been presented

Seg length	Max (bits)	Min (bits)	Average (bits)	Std Dev (bits)
1	185267	476	15598.3	18165.1
2	345123	1186	31197.5	31254.7
4	665058	4741	62395.0	57140.6
8	1053294	10561	124790.0	112555.3
12	1378253	16571	187196.1	167277.3

Table 2: Parameters of frame and segment distribution

Unformatted capacity	11,700 MBytes
Formatted capacity	9,100 MBytes
Interface	Fiber channel dual port
Track-to-track seek (r/w)	0.6/1.1 ms
Average seek (r/w)	8/9.5 ms
Maximum seek (r/w)	19/20 ms
Spindle speed	7200 rpm
Average latency	4.17 ms
Internal transfer rate	80 to 124 M bits / sec
Disks/data surfaces	10/20
Servo heads	embedded
Bytes per sector	512
Bytes per track	78540 to 122173
Sectors per drive	17,773,440
Cylinders	5272

Table 3: Seagate ST19171FC Barracuda-9 hard drive

using the frame grouping scheme of [11], similar results can be obtained using any other scheme. Table 2 tabulates the maximum, minimum, average and standard deviation of the number of bits per segment.

The parameters for the Seagate ST19171FC Barracuda-9 3.5 inch hard drive, which is used here, are shown in Table 3. The disk drive is modelled as follows [10]. The disk seek time varies with the number of traversed cylinders, n, as :

$$seek(n) = \begin{cases} a + b \times \sqrt{n} & \text{if } n < \frac{1}{3} \times n_{max} \\ c + d \times n & \text{otherwise} \end{cases}$$

where a, b are computed using minimum and average seek time values, while c, d are computed using average and maximum seek time values. We have $a = 0.4191442$, $b = 0.1808558$, $c = 3.5012802$ and $d = 0.00256045$.

The rotational latency is taken to be $U[0, \frac{60}{rpm})$, where U is the uniform distribution and rpm is the spindle speed in rotations per minute. Disk transfer rates vary depending on the track location, with outer tracks having a higher transfer rate than inner ones. The transfer rate is taken to vary linearly with track location.

Let the system contain 1000 videos each lasting 100 minutes when played 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. 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.

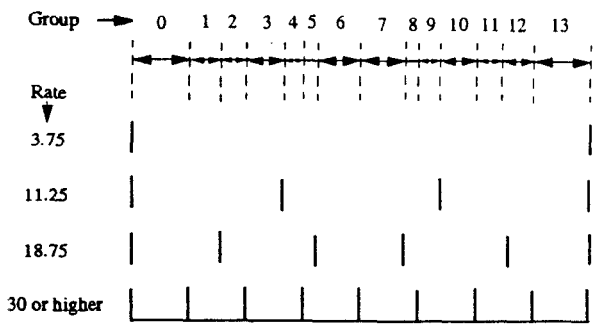

Figure 7: Macro-slots and groups in QuIVeR-1,2,3

5.1 QuIVeR-1,2,3,4

Let the set of playback rates available to the user be : 3.75, 11.25, 18.75, 30, 60, 90, 120 fps. This complies with the constraint of playback rates imposed by QuIVeR.

Let the user rate transition matrix be :

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

Let the average time (in sec) that a user spends viewing at playback rate i be T_i, for $i = \{1, 2, \cdots, 7\}$ and :

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

Solving the balance equations for the resulting markov chain, we get the steady state probability of the user being in a particular state at any arbitrary time to be :

Π = [0.00250643 0.00259562 0.00309465 0.98360655 0.00309465 0.00259562 0.00250643]

Assuming 12 frames/segment and since the slowest rate is 3.75 fps, the round duration is $T_c = 3200$ ms. The macro-slots for segment retrievals of users at different rates is illustrated in Figure 7. In QuIVeR-1,2,3, the number of groups in each service round is 13 and the duration of each of these groups is given in Table 4. In QuIVeR-4, the number of groups in a round depends on the set of playback rates being used in that round as indicated in Table 5. In each of these cases, the number of groups in a round is unaffected by the presence (or absence) of any user at the slowest rate (3.75 fps).

Table 6 compares the performance of the different QuIVeR protocols. The fraction of segments that cannot be retrieved (either partially or completely) as well as the fraction of time the disk subsystem is busy (disk utilization) is tabulated. The former is an indication of the service quality offered to the user. As expected, the performance

Gr. #	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Dur. (ms)	400	240	160	266.7	133.3	80	320	320	80	133.3	266.7	160	240	400

Table 4: Group duration for QuIVeR-1,2,3

Rate(s)	≥ 30	18.75, ≥ 30	11.25, ≥ 30	11.25, 18.75, ≥ 30
# Groups	8	12	10	14

Table 5: # groups in a round for QuIVeR-4

of QuIVeR-2 is better than that of QuIVeR-1, and that of QuIVeR-3 is better than QuIVeR-2. The improvement in performance is not significant. However, a significant performance improvement is observed when QuIVeR-4 is employed. This is due to the fact that a user, for the most part, is viewing the video at the normal rate where the number of groups in a round is eight. The tradeoff between an increase in disk utilization and the corresponding decrease in service quality can also be seen.

For the case of 12 frames/segment and 50 users/disk subsystem, the variance of the average size of a group for QuIVeR-3 was 171.878383.

5.2 QuIVeR-5

QuIVeR-5 imposes additional restrictions on available playback rates than that imposed by QuIVeR-1,2,3,4. Let the set of playback rates available to the user be : 3.75, 7.5, 15, 30, 60, 120, 240 fps. The user model is similar to that for QuIVeR-1,2,3,4. Assuming 12 frames/segment and since the slowest rate is 3.75 fps, the round duration is $T_c = 3200$ ms. The macro-slots for segment retrievals of users at different rates is illustrated in Figure 8. The duration of each of the eight groups is $\frac{3200}{8} = 400$ ms.

The performance of QuIVeR-5 employing *Procedure Q5-A* and *Procedure Q5-B* was almost identical. Hence, we will not distinguish betweeen the two henceforth. Table 6 includes the performance of QuIVeR-5 for a varying number of users in the system. QuIVeR-5 is seen to perform significantly better than QuIVeR-4. This is attributed to the uniform group durations in all rounds and the virtually negligible variance in number of segment retrievals per group. For the case of 12 frames/segment and 57 users/disk subsystem, the variance of the group size (considering 1000 groups of each type) was 0.295477 while the variance of the average size between different group types was 0.017319.

As the number of users in the disk subsystem increases, the average number of retrievals per group increases linearly. Such an increase causes a decrease in the average seek time per retrieval as indicated in Figure 9. Since the average latency and data transfer time are independent of the number of retrievals in a group, the total time taken to retrieve a segment varies in a similar fashion to the seek time. An increase in the number of retrievals per group could result in an insufficient time for all retrievals. An increase in number of users results in an increase in the disk utilization (which is the fraction of time that the disk drive is busy). These variations can be observed in Table 6.

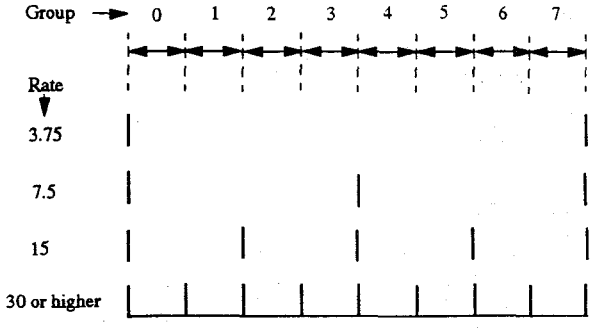

Figure 8: Macro-slots and groups in QuIVeR-5

6 Conclusions

In this paper, we presented the QuIVeR protocol for video retrieval from disk-array based video servers. The set of offered fast playback rates can be arbitrarily chosen while that of slow rates can be chosen from a wide range of playback rates. Guarantees for the existence of a schedule were given. The service quality offered to users (fraction of segments that cannot be retrieved) was examined by simulation using data from the MPEG-1 encoded *Star Wars* video. It was observed that minimizing variance in group duration as well as in the number of segment retrievals assigned to groups greatly improves performance. For the set of playback rates considered for QuIVeR-5 in the numerical examples, it was seen that 52 users could be supported per disk subsystem (storing 30 or 31 videos in 10 disks) with a segment retrieval failure probability of less than 10^{-6}. With 33 disk subsystems needed to store 1000 videos, a total of $33 \times 52 = 1716$ users can be supported by the server with a high service quality. The dedicated set-top box buffer required by QuIVeR is extremely small (compared to other schemes in the literature) and merely needs to be able to store two video segments (when $k = 1$). No buffer is needed at the server. The proposed retrieval scheme, due to its implementation simplicity and high efficiency, is well suited for a real-time application such as interactive VOD.

# Usr	Fraction of unretrieved segments					Disk utilization				
	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5
50	0.00614	0.00570	0.00549	0.00052	0.000000	0.8487	0.8488	0.8488	0.8030	0.7944
51	0.00895	0.00828	0.00808	0.00059	0.000000	0.8614	0.8616	0.8618	0.8161	0.8089
52	0.00990	0.00924	0.00890	0.00069	0.000000	0.8745	0.8749	0.8750	0.8302	0.8216
53	0.01126	0.01061	0.01053	0.00088	0.000002	0.8877	0.8881	0.8882	0.8436	0.8354
54	0.01260	0.01169	0.01133	0.00106	0.000012	0.8999	0.9005	0.9008	0.8576	0.8489
55	0.01666	0.01569	0.01539	0.00139	0.000046	0.9109	0.9116	0.9118	0.8699	0.8620
56	0.02014	0.01917	0.01884	0.00248	0.000090	0.9222	0.9230	0.9232	0.8859	0.8763
57	0.02285	0.02191	0.02153	0.00252	0.000249	0.9341	0.9349	0.9352	0.8977	0.8898
58	0.02759	0.02652	0.02627	0.00395	0.000540	0.9444	0.9453	0.9455	0.9122	0.9041
59	0.03100	0.03021	0.03019	0.00487	0.001257	0.9556	0.9563	0.9563	0.9244	0.9168
60	0.03756	0.03652	0.03618	0.00678	0.002153	0.9625	0.9634	0.9638	0.9360	0.9288

Table 6: Protocol comparison

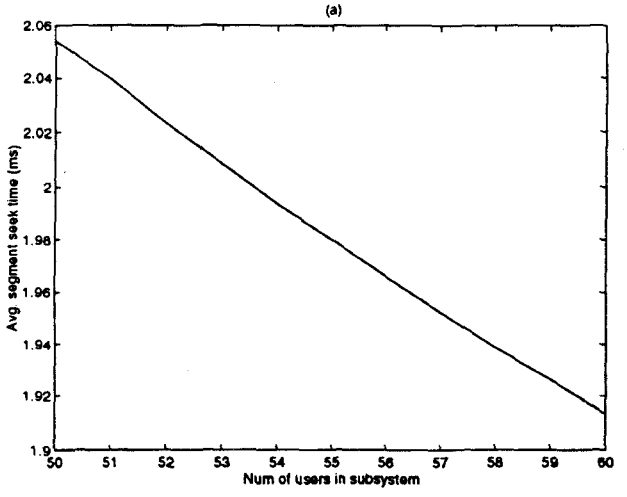

Figure 9: Variation of avg. seek with # in subsystem (Q5)

References

[1] M.S. Chen, D.D. Kandlur, and P.S. Yu. "Optimization of the grouped sweeping scheduling (GSS) with heterogeneous multimedia streams". *ACM Multimedia Conference*, pages 235–242, 1993.

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

[3] D. Deloddere, W. Verbiest, and H. Verhille. "Interactive Video On Demand". *IEEE Comm. Mag.*, pages 82–88, May 1994.

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

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

[6] 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.

[7] V.O.K. Li, W.J. Liao, X. Qiu, and E.W.M. Wong. "Performance model of interactive video-on-demand systems". *IEEE JSAC*, August 1996.

[8] S. Ramanathan P. V. Rangan, H. M. Vin. "Designing an On-Demand Multimedia Service". *IEEE Communication Magazine*, pages 56–64, July 1992.

[9] A.L.N. Reddy and J.C. Wyllie. "I/O Issues in a Multimedia System". *IEEE Computer*, pages 69–74, Mar 1994.

[10] C. Ruemmler and J. Wilkes. "An Introduction to Disk-Drive Modelling". *IEEE Computer*, 27(3):17–28, March 1994.

[11] S. Sengodan and V.O.K. Li. "A Discard-Free Grouping and Retrieval Scheme for Stored MPEG Video". *IEEE Globecom*, 1997. Submitted.

[12] S. Sengodan and V.O.K. Li. "A Quasi-static Retrieval Scheme for Interactive VOD Servers". *Computer Communications*, May 1997.

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

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

[15] H. M. Vin and P. V. Rangan. "Designing a Multiuser HDTV Storage Server". *IEEE JSAC*, 11(1), Jan 1993.