A Shared Buffer Architecture for Interactive VOD Servers*

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}@irving.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. While considering a disk-array based VOD server, use of a shared buffer at the server end may be more economical than the sole use of dedicated buffers at each user's end. In this paper, we propose a simple buffer sharing architecture that may be used when disk-array based video servers are used. Our aim is to support the maximum number of users for a given number of video server disks while employing a simple scheme requiring less buffer space. The number of video segment retrievals that can occur within a certain time (the service round) is maximum when the scan disk scheduling algorithm is used. Consequently, we shall assume use of the scan algorithm for disk retrieval. The VOD server has a buffer manager that directs retrieved segments to appropriate buffer locations depending on their release and deadlines. The release and deadlines of segments are such that buffer requirement at the user's set-top box is minimized to two video segments while avoiding video starvation and buffer overflow at the user's end. We propose a novel scheme for the operation of the shared buffer that aims at increasing buffer utilization and decreasing cell loss due to buffer overflow. An ATM based broadband network is assumed and all segments are stored in buffers as fixed length ATM cells.

1 Introduction

In a Video-on-Demand (VOD) system, all videos are stored in a video server and the user (subscriber) may view any of these videos at any desired time. The user's request (to the video server) as well as the video signal (from the server back to the user) are transported by a broadband communication network such as an Asynchronous Transfer Mode (ATM) network. The user may perform several interactive operations that include pause, stop, slow/fast display and jump forward/backward to any desired location in the video sequence. We believe that reverse play does not have to be provided when a jump backward option is provided.

Since video (as well as audio) is a continuous media, it is delay sensitive. For smooth playback, video segments are needed at regular intervals at a user's end. Failure to deliver video segments when needed results in video starvation and a consequent glitch (or hiccup) in the playback. Since the start and duration of segment retrieval (from disk) as well as network delays are not deterministic, segments may need to be retrieved and transmitted in advance in order to avoid video starvation. These segments arriving at the user, then, need to be buffered until required by the user for display. In this paper, we assume the network is transparent and concern ourselves merely with the video server. The segment arrival at the user, thus, directly relates to its transmission onto the ATM network. The earlier the segment arrival at the user's end (relative to its playback), the larger is the amount of buffer required. Hence, we need to regulate the arrival of segments at a user by regulating their transmission from the server onto the ATM network. One way of achieving this is by regulating the segment retrieval from disk, and transmitting retrieved segments immediately onto the network without any buffering at the server. However, this is undesirable due to the high penalty of disk seek involved. In this paper, we relax the regulation of segment retrieval from disk by employing the scan algorithm, but store these retrieved segments in a buffer at the server and regulate their transmission onto the ATM network.

The choice of buffer location plays an important role in determining the overall cost of the system. Buffer locations affect both the total amount of buffer required as well as the number of users supported by the system. Typically, buffers may be provided at two locations - video server and user's set-top box. Buffers at the server are shared by all users while buffers at a user's set-top box are for the sole use of that user. Consequently, in most cases, use of buffers at the server results in a decrease in the buffers at each user's end and a decrease in the total amount of required buffer.

A lot of work on shared buffers for continuous media appears in the literature [4] [7]. Most of this work has been with regard to video teleconferencing where the time instant when a video frame is generated by each of the sources is

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

0-8186-7780-5/97 $10.00 © 1997 IEEE

known. By shifting the time instants at which different sources generate frames, efficient use of a shared buffer can be made. In the case of a VOD system, however, the time instant when a video segment is retrieved from disk is not deterministic. For instance, when the scan scheduling or grouped sweeping scheduling (GSS) [2] algorithm is used, the order in which segments are retrieved (for the same as well as for different users) is unknown. By permitting such a variable order of segment retrieval, the total disk seek time can be greatly reduced. This is extremely desirable as it permits more number of users in the system. However, in order to regulate the transmission of segments onto the network, a simple and efficient scheme for segment buffering is needed at the server. In this paper, we propose a scheme that aims to achieve this.

The rest of the paper is organized as follows. Section 2 describes the proposed VOD architecture. Section 3 briefly mentions the scheme we adopt for grouping frames into segments. Section 4 describes the disk storage and retrieval issues, while Section 5 deals with the performance evaluation and numerical results. Finally, we conclude in Section 6.

2 VOD system architecture

The VOD system being considered is shown in Figure 1. As seen in the figure, user displays and the VOD server are interconnected by an ATM based high-speed broadband network. The interface of the user (rather, his/her remote) as well as the display to the broadband network is via a set-top box. The set-top box is assumed to have adequate buffer storage as well as circuitry for performing such functions as MPEG decoding.

The VOD server comprises a set of disk subsystems, a retrieval scheme manager, a staging buffer, a buffer manager, a main buffer (comprising several buffer modules) and a time division multiplexer (TDM). Interactive requests from a user are transported by the ATM network to the retrieval scheme manager. In order to maximize the number of supported users, the retrieval scheme used by the retrieval scheme manager is the scan disk scheduling algorithm [1][11]. These retrieved segments pass through the staging buffer prior to being directed by the buffer manager to appropriate locations in the main buffer (subject to availability of buffer space.) Both the staging buffer and the main buffer are shared by all the users and all segments are stored in them as ATM cells. Layered MPEG coding [8] may be used so that each segment comprises high priority (HP) and low priority (LP) cells. However, we will not consider this case any further here.

The storage (location) of segments in the staging buffer may be arbitrary, but that in the main buffer is based on the buffer management scheme being used by the buffer manager. The buffer manager determines the appropriate location (in the main buffer) for retrieved segments based on their release and deadlines for transmission on to the network. The release and deadline of each segment (for a given amount of dedicated buffer at the user's set-top box) depend on the latest time by which the segment is

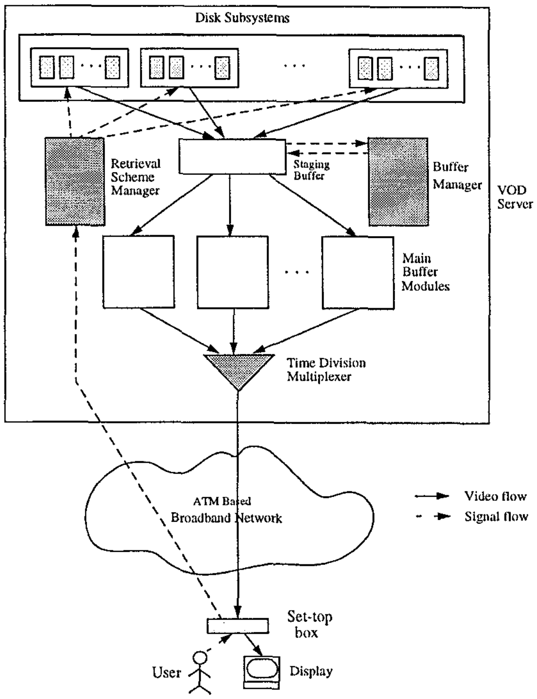

Figure 1: Architecture of VOD system

needed to avoid video starvation. In order to maximize the number of cells that may be transferred to the main buffer (and hence, minimize cell loss at the buffer), the algorithm used by the buffer manager is an Earliest Deadline First (EDF) algorithm. The TDM, then, transmits these ATM cells in a time division multiplexed fashion to the user via the broadband network.

2.1 Retrieval scheme manager

The retrieval scheme manager is responsible for determining the order in which segments required in a service round are retrieved from disk. In other words, the disk scheduling algorithm resides in the retrieval scheme manager. Several disk scheduling algorithms may be used. These include fixed order scheduling, scan scheduling, Grouped Sweeping Scheduling (GSS) [2] and Nonpreemptive Earliest Deadline First (N-EDF) [9] scheduling. The total disk seek time in a service round depends on the particular disk scheduling algorithm being used and is minimized with the scan scheduling algorithm [1]. Since a small total seek time implies that more segments may be retrieved in a round, we shall employ the scan scheduling algorithm in the retrieval scheme manager.

While the scan scheduling algorithm maximizes the number of segments that may be retrieved from disk in a service round, it has the disadvantage that no control exists over the order in which these segments are retrieved. Hence, if n segments are retrieved for a particular user in a service round, then the retrieval of these n segments from disk may be spaced uniformly apart in the service round (most desirable) or they may be retrieved consecutively (least desirable.) If a maximum of n_m segments may be retrieved for any single user in a service round, then lack of any buffer at the server implies that, in the worst case, n_m segments are transmitted onto the network (and arrive at the user) consecutively. Hence, a buffer capable of storing up to n_m segments needs to be provided at each user's set-top box in order to avoid buffer overflow. This is not an efficient way of providing buffer storage since the total buffer required becomes very large and at most times the buffer utilization is extremely low.

2.2 Buffer manager

In order to decrease the total buffer required and increase buffer utilization, the n segments retrieved for a user in a service round need to be transmitted to the user uniformly spaced within the round. If a video segment is taken to be an atomic unit, a lower bound on the buffer requirement at the user's set-top box is one segment. A buffer requirement of one segment at the user's set-top box implies that the transmission of segments for a user should take place (exactly) at required time instants in a service round. When user interactivity is permitted, the user playback rate and hence, the number of segments retrieved for a user in a service round can vary between rounds. Under such a situation, it can be shown [9] that a buffer restriction of one video segment decreases the number of possible segment transmissions in a service round.

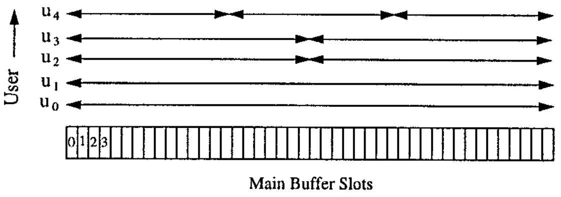

Figure 2: Illustrating user macro-slots and EDF

In order to maximize the number of segment transmissions (on to the ATM network) in a service round, we do the following. For a user requiring n segments in a round, the main buffer is divided into n equal (logical) "macro-slots" [9] and the i-th segment (where $i = 0$ to $n-1$) for the user is stored anywhere (and, possibly non-contiguously) in the i-th macro-slot. With a total of C_b slots in the main buffer, cells associated with segment i of a user requiring n segments need to be stored in slots with numbers between $\frac{i \times C_b}{n}$ and $\frac{(i+1) \times C_b}{n}$. Since different users have different playback rates, the number of macro-slots into which the main buffer is divided varies from one user to another. Macro-slots belonging to users requiring the same number of segments in a round are fully overlapping, while those belonging to users requiring different number of segments in a round may be partially overlapping. This is illustrated in Figure 2 where users u_0, u_1 have one macro-slot each; users u_2, u_3 have two macro-slots each; and user u_4 has three macro-slots. This implies that users u_0, u_1 need one segment in a service round; users u_2, u_3 need two segments; and user u_4 requires three. The numbering of cell slots comprising the buffer is according to the order in which the TDM accesses them.

Since segments belonging to different users need to be transferred from the staging buffer to contending locations in the main buffer, the order in which these segments are transferred is important. The transfer needs to take place in such a fashion that any resulting cell loss due to lack of space in the main buffer is minimized. An EDF algorithm is used for this purpose and is as follows :

1. Let **U** be the set containing all macro-slots.

2. Select the macro-slot with the earliest deadline. Transfer cells of segments which are assigned this macro-slot (in any order) to the cell slots in the buffer within this macro-slot.

3. If all cell slots comprising this macro-slot are filled, then drop any remaining cells that do not have storage space in this macro-slot. Go to step 5.

4. If all cells belonging to all segments assigned to this macro-slot have been transferred, go to step 5.

5. Delete this macro-slot from **U**.

6. Go to step 2 if **U** is non-empty. Else, stop.

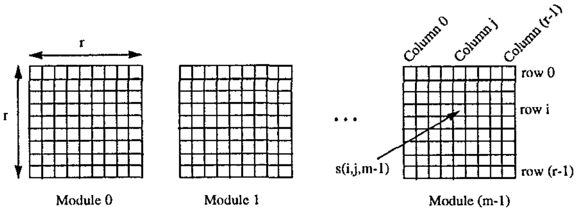

Figure 3: Main buffer

For any distribution of empty slots in the main buffer, the above algorithm minimizes cell loss during transfer. However, the distribution of empty slots greatly affects the cell loss. In the following subsection, we present a novel scheme that aims at obtaining a distribution of empty slots that minimizes cell loss.

2.3 Buffer architecture

Let C_b denote the total number of cell slots in the main buffer. As shown in Figure 3, the main buffer comprises m buffer modules. Each module comprises r^2 cell slots that are (logically) arranged as a square with r rows and r columns. We shall denote the slot corresponding to row i, column j of module k as $s(i, j, k)$, where $i, j = 0, 1, 2, \cdots, r-1$ and $k = 0, 1, 2, \cdots, m-1$. We also have : $C_b = m \times r^2$.

Associated with each cell slot $s(i, j, k)$ is a slot number, whose value depends on whether the current service round is odd-numbered or even-numbered. For even-numbered service rounds, slot $s(i, j, k)$ is assigned a number $(irm + jm + k)$. For odd-numbered service rounds, slot $s(i, j, k)$ is assigned a number $(jrm + im + k)$. Thus, cell slots in different modules with the same values of i (row number) and j (column number) are numbered consecutively. In even-numbered service rounds the slots of each module are numbered row by row, while in odd-numbered rounds they are numbered column by column. Assignment of slot numbers in this fashion results in the creation of empty slots in a desirably "uniform" manner when the TDM empties the slots (as seen later). As mentioned earlier, the macro-slots of users are based on the slot numbers and since the order of segment retrieval by the scan algorithm is random, such a "uniform" creation of empty slots is desirable for a decrease in cell loss.

At the start of any service round i, transmission of cells stored in the buffer modules onto the ATM network begins in a TDM fashion. The TDM accesses the slots in the increasing order of their slot numbers corresponding to the previous round (i.e., round $(i-1)$). This is because segments transmitted in round i are those that were retrieved (and stored in the buffer) in round $(i-1)$. Simultaneously, new segments are being retrieved from disk and need to be buffered. These new segments are stored initially in the staging buffer. When the staging buffer is full, its contents are transferred to the appropriate locations in the main buffer. Let the size of the staging buffer be denoted by C_s. For the same parameters of retrieved segments, the amount of cell loss in the main buffer decreases as the value of C_s increases. At an extreme, when C_s equals C_b and if the size of all retrieved segments does not exceed C_b, the cell loss during transfer to the main buffer (which occurs only once in a round) goes to zero. However, in order to decrease the total buffer space, C_s is usually a small fraction of C_b. Thus, even when the total size of all retrieved segments does not exceed C_b, some cell loss can result during their transfer to the main buffer.

At the end of round i, all segments retrieved from disk during round i will have been stored in the main buffer (with some loss) and all segments that had been stored in the main buffer at the end of round $(i-1)$ will have been transmitted on to the network. The staging buffer is empty at this point.

If C Mbps is the channel capacity (at the output of the TDM) and C_b the total number of cell slots in the main buffer, then the time taken by the TDM to complete one service round is $53 \times 8 \times C_b/C$ μs. As we will see later, with n_f frames per segment and r_0 fps as the slowest playback rate, the duration of the service round equals n_f/r_0 sec. Equating the two, we have $53 \times 8 \times C_b/C = (n_f/r_0) \times 10^6$. Thus, the channel capacity is determined as $C = \frac{53 \times 8 \times C_b \times r_0}{n_f \times 10^6}$ Mbps.

3 Grouping frames into segments

During video playback, the unit of video information that needs to be displayed at regular intervals is a video frame. However, the popular coding scheme for videos is MPEG coding [8] which uses interframe coding in addition to intraframe coding. Decoding of a frame, in such a case, may depend on previous as well as subsequent frames. Consequently, arrival of a frame at a user before its playback deadline is not sufficient, but all frames needed for the decoding of a particular frame need to arrive at the user prior to the deadline of that frame. Thus, it is advantageous to group frames into segments [3]. Arrival of segments at the user is such that frame decoding takes place in a causal manner. Since each segment comprises the same number of video frames, segments need to arrive at regular intervals at a user's end.

In [3], a segment is taken to comprise the set of frames from one I frame up to (and excluding) the next one. When the number of segments retrieved in a service round is proportional to the user's playback rate, decreasing the segment size decreases the duration of a service round. A decrease in the duration of the service round is extremely desirable in a quasi-static retrieval scheme such as ours (as described later) as it results in a decrease in the interactive latency. In addition, skipping segments for fast playback requires dropping (i.e., not used for playback) some frames in retrieved segments when segments are grouped as in [3]. This is because other frames necessary to decode these (dropped) frames are present in skipped segments. Due to these reasons, we introduce a different scheme for grouping frames into segments which does not suffer from the drawbacks mentioned. The scheme is described in detail in [10] but the main idea is highlighted here.

Let the number of B frames between any two consecutive I/P frames be denoted by n_B. Let the number of frames per segment be n_f. Sets of n_f consecutive I/P frames and sets of n_f consecutive B frames constitute a segment. The segment numbering is such that between any consecutive I/P segments, there exists exactly n_B B segments. With such a segment creation scheme, decoding of a frame belonging to any segment only requires frames belonging to earlier segments. When a fast playback is desired, segments have to be suitably skipped. Segment skipping needs to be such that no frames belonging to retrieved segments have to be discarded (for want of other frames needed for their decoding). Moreover, the display quality has to be within a desirable one. Techniques for segment skipping that achieve these two purposes are detailed in [10].

Hence, for our purpose, there exists a grouping scheme that can accommodate any desired number of frames per segment. During fast playback, techniques exist for selecting segments to be retrieved so that no frames belonging to retrieved segments have to be discarded (due to unavailability of other frames needed for decoding).

4 Disk storage/retrieval issues

Let the total number of disk subsystems in the VOD server be N_{ds} and the total number of disks in the system be N_d. The number of disks per subsystem is N_{ds}/N_d. We assume that a segment is striped across all disks in a subsystem. Striping segments across the disks of a subsystem results in balancing the load among disks of a subsystem. In addition, since at any given time all the disks of a subsystem are retrieving (fragments of) the same segment, the effective data transfer rate of the disk subsystem is N_{ds}/N_d times that of a single disk. While the value of N_d depends on the total number of videos to be stored, the value of N_{ds} depends on factors such as available memory and desired user QoS [9].

Each disk subsystem may be independent and hold a video in its entirety, or alternatively, different segments of the same video may be stored on different disk subsystems. Irrespective of what the case may be, the concept of service rounds exists for a disk subsystem within which a certain number of segments need to be retrieved. The service rounds associated with different disk subsystems begin and end at the same time. We will assume that the total load is balanced among the different disk subsystems for all service rounds. Hence, any performance analysis can be limited to one disk subsystem.

Let the set of playback rates that the VOD system offers the user be $\{r_0, r_1, \cdots, r_{N_r-1}\}$, where N_r is the total number of playback rates offered to the user. Without loss of generality, we assume that $r_i < r_j \; \forall \; i < j$. We will consider the duration of a service round to equal the time taken to play back a video segment at the slowest rate (r_0). If a segment contains n_f frames, then the round duration equals n_f/r_0 sec. In addition, we will also assume that all offered playback rates are integral multiples of r_0. With these requirements, each active user in the system requires at least one video segment in each service round and the number of segments needed is an integer (which are desirable features).

4.1 Number of retrieved segments

Consider a user viewing a video at a playback rate of r_{ui} fps, where $i = \{0, 1, \cdots, N_r - 1\}$. Since the number of segments retrieved for a user in any service round depends on the user's playback rate, we shall denote it by the function $f(r_{ui})$. It is desirable from a disk load and network congestion viewpoint to suitably skip segments during fast playback [3]. In such cases, the number of retrieved segments is proportional to the user's playback rate for normal and slow playback, but equals that for the normal playback during fast playback. If r_n is the normal playback rate, we have :

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

4.2 Startup and interactive latency

Since the operation is quasi-static, startup and interactive (rate change, pause/stop, resume, jump) requests made after the start of a round are batched until the start of the next round.

We saw earlier that a macro-slot of any user in any service round encompassed a certain number of cell slots in the main buffer. These cell slots need to store one segment of that user. A segment that is retrieved in one macro-slot is played back during the next. Hence, startup and jump requests made during a round have to be batched till the end of the round, suffer from an additional delay of T_c due to the cell transfer into the main buffer and, finally, take effect at the end of the first macro-slot of the next round. 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 round after the next (and not at the end of the first macro-slot of that round). Table 1 tabulates the latency due to various user requests. Here, k is the number of segments that have to be prefetched prior to start of playback. This parameter depends on the nature of grouping frames into segments. For the method of grouping frames into segments as described in [10], $k = 1$.

5 Performance analysis and results

5.1 Storage preliminaries

The data that we use is based on the MPEG encoded video "Star Wars" [6]. We will assume that this video is representative of all videos in the disk system. We assume that frames are grouped into segments as mentioned earlier. The ratio of the number of I, P and B frames for this encoding was 1 : 3 : 8. Some parameters of the frames

Interaction	Latency[1]
Start	$U[0,T_c) + T_c + kT_c(r_n/r_0)$
Jump	$U[0,T_c) + T_c + kT_c(r_n/r_0)$
Pause/stop	$U[0,T_c) + T_c$
Resume	$U[0,T_c) + T_c$
Change rate	$U[0,T_c) + T_c$

Table 1: Latency of user requests

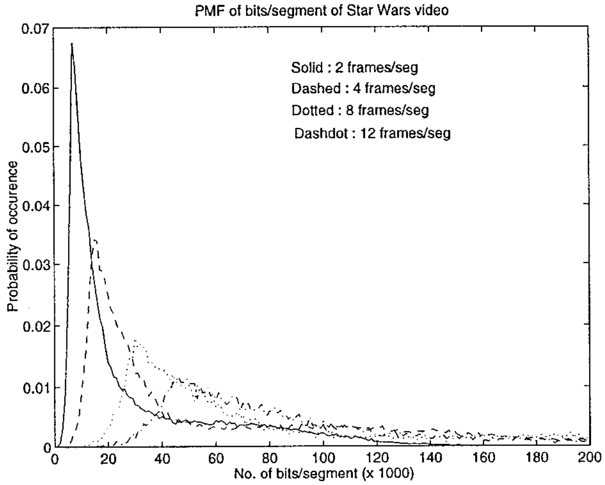

Figure 4: Probability mass function of bits/segment for the "Star Wars" video

and the resulting segments are indicated in Table 2 where n_f denotes the number of frames per segment. Figure 4 plots the probability mass function (pmf) of the number of bits/segment.

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.

Let us assume that the possible playback rates (in fps) are:

$r_0 = 10$, $r_1 = 20$, $r_2 = 30$, $r_3 = 40$, $r_4 = 50$, $r_5 = 60$, $r_6 = 90$

For the set of playback rates indicated, the number of segments retrieved in a round for a user is given by: $x_0 = 1$, $x_1 = 2$, $x_2 = x_3 = x_4 = x_5 = x_6 = 3$

Since the duration of a service round equals the amount of time a segment would last a user at rate r_0, (for n_f frames/segment) we have :

T_c = service round duration = 100 × n_f ms

Let the parameters of the main buffer be $m = 17$, $r = 100$ resulting in a total buffer storage for 170000 cells. Let the staging buffer capacity be $\frac{1}{10} - th$ of the main buffer.

n_f	Max (bits)	Min (bits)	Avg (bits)	Std Dev (bits)
1	185267	476	15598.33	3.3×10^8
2	345123	1186	31197.49	31254.73
4	665058	4741	62394.98	57140.61
8	1053294	10561	124789.96	112555.24
12	1378253	16571	187196.05	167277.25

Table 2: Parameters of frame and segment distribution

5.2 User model

Let N_r denote the total number of playback rates that the VOD system offers the user. As far as load on the disk (i.e., number of segment retrievals) is concerned, the user can be in one of $(N_r + 1)$ possible states - the N_r playback rates and the pause/stop state. In order to consider the operation during heavy load, we shall ignore the pause/stop state and only consider the case where the user changes his/her playback rate. We assume that the behaviour of each user is independent and identically distributed (i.i.d.). Let the transition probability matrix for the user states be :

$$P = \begin{pmatrix} 0 & 0.3 & 0.5 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.3 & 0 & 0.5 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.15 & 0.2 & 0 & 0.2 & 0.15 & 0.15 & 0.15 \\ 0.05 & 0.05 & 0.5 & 0 & 0.3 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.5 & 0.15 & 0 & 0.2 & 0.05 \\ 0.05 & 0.05 & 0.5 & 0.05 & 0.15 & 0 & 0.2 \\ 0.05 & 0.05 & 0.5 & 0.05 & 0.15 & 0.2 & 0 \end{pmatrix}$$

where each 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 an interactive request.

The amount of time spent by a user in a particular state is assumed to be exponentially distributed. Hence,

$T_i = \frac{1}{\mu_i}$ = mean sojourn time for user in state i; where $i = 0, 1, \cdots, m-1$

Let $[T_1\ T_2 \cdots T_7]$ = [5 5 600 10 10 10 10]

In order to determine the probability that a user is in a particular state at any given time, we need to transform the user state transition chain into a markov chain. The resulting markov chain has m states where $\forall i, j = 0, 1, \cdots, m-1$:

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

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

Π = [0.0026 0.0029 0.9731 0.0052 0.0061 0.0054 0.0046]

Associated with each user i, let us define a discrete random variable (RV) $x_i = f(r_{ui})$. Here, r_{ui} is the playback rate of user i and $f(r_{ui})$ is the number of segments retrieved in a service round for a playback rate of r_{ui}. Since

the behaviour of each user is assumed to be independent and identical, the RV's x_i are i.i.d..
We have: Mean(x_i) = 2.992; Variance(x_i) = 0.0134

5.3 Cell loss at disk system

A large load on the disk system may result in cell loss during retrieval. This loss is due to the fact that the end of a service round has been reached but some segments are yet to be retrieved. Assuming a known number of users in the system, we shall now determine the cell loss during segment retrieval from disk. In order to simplify the analysis, we will assume that the load is balanced among the various disk subsystems during all service rounds. Consequently, we only need to analyze the effect that a single disk subsystem with k users has on the cell loss probability. This value, then, equals that due to $N_{ds} \times k$ users in the entire system.

Let there be k users currently in a disk subsystem. Define an RV $X_k = \sum_{i=0}^{i=k-1} x_i$, i.e., X_k represents the total number of segments retrieved (by all k users) in a service round and is an RV that equals the sum of k i.i.d. RV's. The mean (η') and variance (σ'^2) of the distribution of X_k is thus $k\eta$ and $k\sigma^2$ respectively. Since k is a large number, the central limit theorem states that the distribution of X_k approximates a Gaussian distribution with mean η' and variance σ'^2.

Let y_i be an RV denoting the size of the i-th retrieved segment in a service round. Let $n = X_k$ segments be retrieved totally (for all k users) in a service round. Define an RV $Y = \sum_{i=0}^{i=n-1} y_i$, i.e., Y denotes the total size (in bits) of all retrieved segments in a service round. There exists a correlation between the segment size (RV's y_i) associated with the same user, but that associated with different users are independent and identically distributed. Since the total number of users in the system (k) is much larger than the maximum number of segments retrieved for any single user, we may again use the central limit theorem to approximate the distribution of Y by a Gaussian distribution. If η_1 and ${\sigma_1}^2$ denote the mean and variance of the segment size distribution, then the (Gaussian) distribution of Y has a mean of $\eta' \times \eta_1$ and a variance of $\eta' \times {\sigma_1}^2 + (\eta_1)^2 \times \sigma'^2$ [5].

Figure 5 depicts the cumulative distribution of X_k for two values of k. With $k = 650$, the average number of cells generated in a service round is nearly 159000. (The total number of available cell slots in the main buffer is 170000.) For illustration, we also consider an extreme case of $k = 1000$ users, where the average number of cells that need to be retrieved in a round is much larger than that which can be retrieved. The latter case would only affect the cell loss during disk retrieval but not that due to transfer to the main buffer. Figure 5(a) depicts the case with $k = 650$, while Figure 5(b) depicts the case with $k = 1000$. For the case with $k = 650$, the total number of required segment retrievals in a cycle varies between 650 (when all users have the slowest playback rate) and 5800 (when all users have the fastest playback rate); for the case with $k = 1000$, this value varies between 1000 and 9000.

Figure 6 depicts the cumulative distribution of Y for the two values of k. Figure 6(a) depicts the case with $k = 650$, while Figure 6(b) depicts the case with $k = 1000$. Using these figures, we can determine the probability that the total number of bits to be retrieved in a round exceeds a certain value. In other words, we can obtain statistical results for the probability of cell loss during disk retrieval.

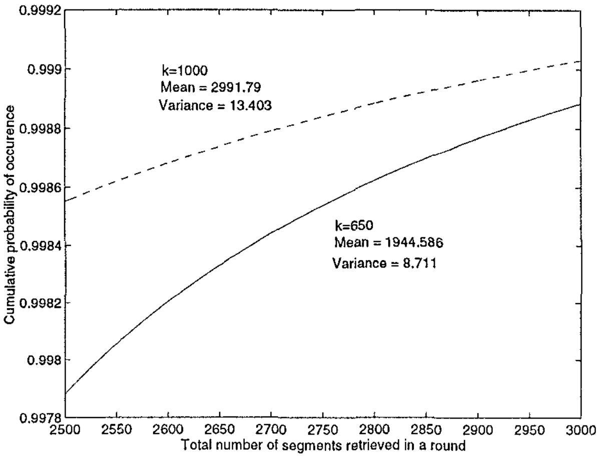

Figure 5: Distribution of total number of retrieved segments in a round for 650 (solid) and 1000 (dashed) users

5.4 Cell loss at buffer

Video segments stored in the staging buffer are transferred to the main buffer periodically. We consider the case where this transfer occurs ten times during a service round. Hence, the transfer occurs at times $i \times (T_c/10)$ from the start of the service round, where $i = \{1, 2, \cdots, 10\}$. This implies that the number of video segments that the staging buffer needs to store equals that retrieved within a time duration of $T_c/10$. As seen earlier, the value of T_c chosen was $100 \times n_f$ ms since $r_0 = 10$ fps. With 650 users using the disk subsystem and $n_f = 2$, simulations were carried out to determine the cell loss due to unavailability of storage space in the main buffer.

Simulations carried out with the parameters described resulted in a cell loss of 9003 cells among 902121 cells that passed through the staging buffer in six rounds. The maximum number of cells that can be stored in the main buffer in six rounds equals 6 × 170000 = 1020000. Thus, the utilization of the main buffer was 88.44%, for which the cells dropped due to lack of space at the main buffer was 1%.

6 Conclusion

In this paper, we proposed a new buffer sharing architecture that may be used for an interactive VOD system. Early simulations carried out resulted in a cell loss during transfer to the main buffer of 1% when the buffer was 88.44% utilized. The operation of the main buffer results in an efficient use of the buffer and in a low cell loss. Sta-

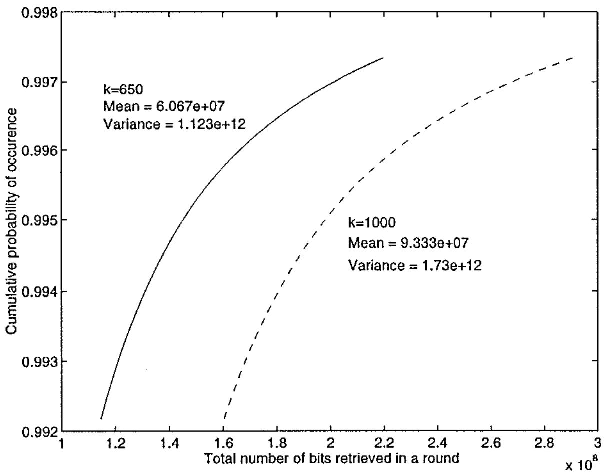

Figure 6: Distribution of total number of retrieved bits in a round for 650 (solid) and 1000 (dashed) users

tistical results for the cell loss during disk retrieval were presented. Since the interactive latency depends on the duration of the service round which depends on the size of retrieved segments, a new scheme for grouping frames into segments was used.

We are currently working on several issues that affect the system performance, some of which are :

1. For the same main buffer space (mr^2), the choice of the number of modules (m) and the dimension of each module (r) affects the nature in which empty slots are created in the buffer. The effect of a variation of these parameters on the cell loss needs to be modelled.
2. The option of not channeling some retrieved segments through the staging buffer could result in a smaller cell loss. For instance, at the start of the round, retrieved segments can be directed to the staging buffer until it is full. Any segments retrieved from this point onwards can be sent directly to the appropriate locations in the main buffer. At the end of the round, the contents of the staging buffer can be transferred to the main buffer using the N-EDF algorithm.
3. The retrieval scheme used in this paper was a quasi-static one where all user interactions take effect only at the start of service rounds. A dynamic retrieval scheme where the effect of a user interaction takes effect earlier may be desirable as it would decrease startup/interactive latencies.

We shall provide comprehensive results on these issues shortly.

References

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

[2] Ming-Syan Chen, Dilip D. Kandlur, and Philip S. Yu. "Optimization of the grouped sweeping scheduling (GSS) with heterogeneous multimedia streams". *Proceedings of the ACM International Conference on Multimedia*, pages 235–242, 1993.

[3] 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.

[4] Shyamala Chowdhury and Kazem Sohraby. "Alternative Bandwidth Allocation Algorithms for Packet Video in ATM Networks". *IEEE INFOCOM*, pages 1061–1068, 1992.

[5] Alvin W. Drake. "Fundamentals of Applied Probability Theory". *McGraw Hill Book Company*, pages 109–112, 1967.

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

[7] Dwight J. Makaroff and Raymond T. Ng. "Schemes for Implementing Buffer Sharing in Continuous-Media Systems". *Information Systems*, 20(6):445–464, 1995.

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

[9] Senthil Sengodan and Victor O.K. Li. "A Quasi-static Retrieval Scheme for Interactive Disk-Array Based Video Servers". *ICCCN*, pages 3–8, October 1996.

[10] Senthil Sengodan and Victor O.K. Li. "A Discard-Free Grouping and Retrieval Scheme for Stored MPEG Video". *IEEE ICC*, 1997. Submitted.

[11] 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.