A Scalable Video-on-Demand Service for the Provision of VCR-Like Functions[1]

H.J. Chen, A. Krishnamurthy, T.D.C. Little, and D. Venkatesh,
Boston University

Multimedia Communications Laboratory
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877, (617) 353-6440 fax
tdcl@bu.edu

MCL Technical Report 03-01-1995

Abstract– One approach to supporting continuous media traffic is via resource reservation at connection establishment time. The shortcomings of this approach are the lack of flexibility in re-negotiating bandwidth parameters after connection establishment and the inability to react to changing server load conditions. In this paper we propose a "selected access scheme" for re-negotiating continuous media delivery from a video-on-demand server. We consider the overall system perspective on resource availability rather than a per-connection resource management scheme. The selected access scheme is combined with data placement strategies for the server to dynamically access desired video frames from a shared storage device. This approach is especially suitable for video data compressed and stored using an inter-frame encoding scheme such as specified by the MPEG standard. Simulation results show that the proposed model can utilize server bandwidth, improve the reliability of playback, reduce the buffering needed, and support VCR-like functions without blocking user access to the storage device.

Keywords: Selected access scheme, data placement strategies, scalable service, MPEG, quality of service and VCR-like functions.

[1]In *Proc. 2nd Intl. Conf. on Multimedia Computing and Systems*, Washington D.C., May 1995, pp. 65-72.

1 Introduction

Rapid advances in storage and networking technology have enabled the support of video-on-demand applications using computer networks. Such applications are characterized by their real-time nature and require a wide range of performance guarantees from the components involved in the end-to-end transfer of data. Due to the large volume of data involved (e.g., video) and the real-time nature of their retrieval, new techniques are required for the efficient use of network and storage bandwidth, and the provision of interactive operations at an acceptable quality of service.

Loosely, quality of service (QOS) can be associated with the following parameters: picture resolution, frame loss over time, the distribution of dropped frames, response time, etc. A common approach to provide guarantees on QOS is to reserve sufficient resources for data transfer at the time of session establishment. Storage resources are then scheduled among different sessions in a manner that guarantees the requirements of individual sessions. Most proposed resource allocation policies are static in nature; allocations made during connection establishment are constant through the lifetime of a connection. The disadvantages of such approaches, in contrast to dynamic resource allocation schemes in which reservations can be changed while the session is in progress, include lack of scalability or graceful degradation and the inability to support some interactive operations such as fast-forward on video.

In this paper we propose a mechanism for the storage and retrieval of MPEG-I or MPEG-II encoded video that supports VCR-like functionality while providing QOS guarantees through dynamic resource reservation.[2] The ability of a video file system to meet the timing requirements for multiple sessions is limited by the unpredictable nature of disk seek latencies [11]. Due to the time-varying characteristics of compressed video it is difficult to predict the disk production and the display consumption rates. Scheduling is even more critical for inter-frame encoding schemes such as specified by MPEG as the loss of a small portion of data in retrieval can lead to a significant increase in the number of frames that are dropped at the time of playout. Furthermore, it is desirable to support interactive access to video (e.g., *pause, reverse, fast-forward* and *fast-reverse*) which further complicates scheduling.

We perceive two possible schemes for the retrieval and delivery of MPEG-encoded video from a physical storage device. These are *frame retrieval* and *block retrieval*.

Frame retrieval: In this scheme, each video frame is retrieved as an independent entity

[2]We consider true-VOD servers with read-only VCR-like functionality, and assume that no editing capabilities are provided.

and the server can dynamically access video frames by their temporal relationships. This scheme is appropriate for retrieving video from random access storage or from a single-user file system. For example, in a disk with a 30 Mb/s bandwidth and an average seek time of 15 ms, the disk latency is 450 ms per second for a frame access to 30 frame/s video stream. However, the data transfer time is only 50 ms for the same traffic at a rate of 1.5 Mb/s without frame access. The resulting overhead is 900 %. The frame retrieval scheme is infeasible for large scale multiple access file systems due to the size of disk seek latencies.

Block retrieval: Here, the file server retrieves data by switching service among sessions. The file system achieves a high disk bandwidth utilization by assigning long disk retrieval times, thereby sharing the seek and latency delays among a large number of frames read from the disk. Unfortunately, it is difficult to provide scalable services with this scheme (e.g., scale down the frame rate from 30 to 15 frame/s) and it is inefficient for interactive operations such as fast-forward.

The proposed "selected access scheme" is a modification of the block retrieval scheme which provides the flexibility of the frame retrieval scheme. With this approach, the server can dynamically access desired video frames from a shared storage device and can provide VCR-like functions. The approach permits the disk to switch among tasks to obtain statistical multiplexing gains and for the scheduler to dynamically change the bandwidth allocated to individual sessions. The context of this work is to design a video-on-demand server that can satisfy a variety of objectives that include utilizing the storage bandwidth, minimizing the total cost, providing virtual VCR capabilities, and still achieve an acceptable quality of service.

The remainder of this paper is organized as follows. In Section 2 we investigate the characteristics of MPEG-encoded video data including decoding dependencies and time-varying behavior and develop a physical disk scheduling scheme which supports multiple real-time temporal data streams. In Section 3 we describe an alternate storage scheme for MPEG-encoded data and a "selected access scheme" for retrieving data. In Section 4, we describe how our selected access scheme can support VCR-like functions. We discuss related work in Section 6. Section 7 concludes the paper.

2 Retrieval Model for MPEG Video Streams

We now consider the inter-frame dependencies of the MPEG encoding scheme and their effect on disk scheduling. In MPEG, frames are grouped using a pattern that is typically fixed at the time of encoding. Fig. 1 shows a sequence of MPEG encoded I, P and B-frames for a group size of 10. Frames within a group are considered to be a logical unit. As a result, losses are sensitive to the particular frame that is corrupted or dropped. For example, if the beginning I frame in a sequence is lost, all frames within its group must be discarded as they are dependent on the I frame for decoding. As a result of this dependency, frame losses in MPEG-encoded video are correlated.

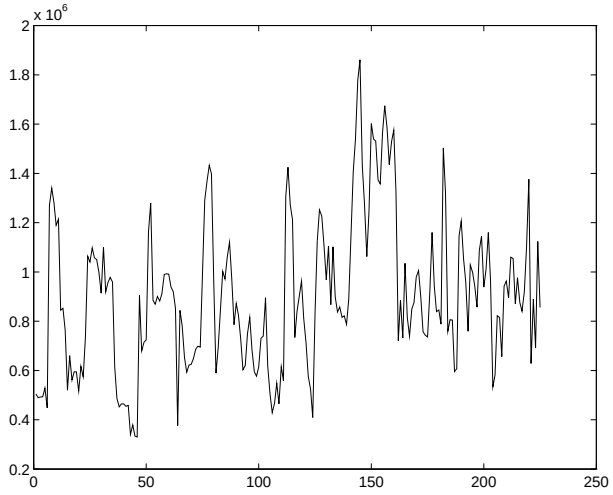

Figure 1: MPEG Frame Dependencies

The MPEG compression algorithm also introduces a highly time-varying bandwidth characteristic [6, 12]. The size of a video frame depends on the encoding algorithm, frame type, and the activity within the video sequence. Fig. 2 shows the characteristics of the video traffic for 2400 frames (80 seconds) of video. In this example, the video source generates 30 frame/s with an aspect ratio of 320×240 pixels. The average rate μ_λ over all 2400 frames is 0.6369 Mb/s and the standard deviation σ_λ for the traffic rate is 0.2305 Mb/s with a sampling interval τ of 10 frames.

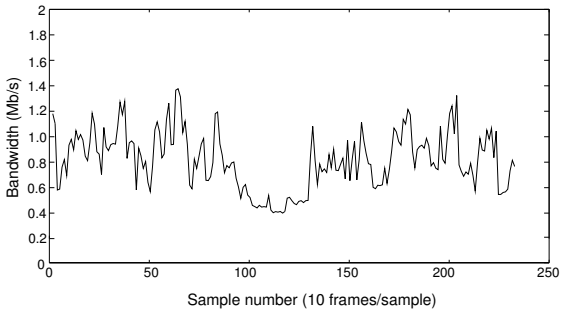

Figure 2: Time-Varying MPEG Video Traffic

2.1 Disk Access Scheduling and Bandwidth Requirements

In this section we describe the scheduling constraints for the acceptance of a set of multimedia sessions and the corresponding disk bandwidth requirements. For demonstration purposes we use the round-robin service scheme to support real-time multiple stream retrieval. We predefine a fixed length of working period T_{period} for a set of multimedia tasks as shown in Fig. 3. However, the proposed approach is not limited to the round-robin scheme and can accommodate alternative mechanisms such as GSS or SCAN to achieve a higher disk utilization [4].

During a working period, the scheduler switches among all active multimedia sessions [3]. It retrieves the exact amount of data necessary for each session to ensure continuous playout from the buffer. This keeps each session busy until the next retrieval period when the buffer is ready to be refilled. (Buffer management policies to ensure continuous playout are described elsewhere [7, 11].) If session i displays m frames per second, the file system must read exactly $m \times T_{period}$ frames for session i in the working period. Let $S(i)$ be the size of these $m \times T_{period}$ frames. If R is the entire disk bandwidth, each session i shares an interval $T(i)$ where $T(i) = \frac{S(i)}{R}$. As shown in Fig. 3, the total interval used for multimedia sessions plus the disk seek latency should be less than the working period T_{period} to account for the variances in disk seek latencies and MPEG traffic volume. In other words, the preset period T_{period} must be greater than the time needed to transfer data from the disk for all sessions.

To prevent starvation, the requirement for the length of T_{period} is:

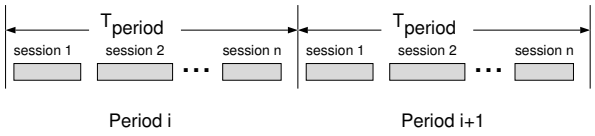

Figure 3: Round Robin Scheduling Assignment

$$T_{period} \geq \sum_{1}^{n} \frac{S(i)}{R} + \sum_{1}^{n} T_{latency}(i) \tag{1}$$

In Eq. 1, n is the number of concurrent sessions from the disk and $T_{latency}(i)$ is the switching latency from session i to $i+1$.

3 Disk Storage Organization and Access Scheme

As described in Section 2, a MPEG-encoded video frame cannot be decoded until the entire frame and any reference frames have completely arrived. Every B-frame needs to reference two other frames (two P-frames, one I and one P-frame, or two I frames). For example, in Fig. 1, frame 5 (B-frame) cannot be decoded without frames 4 and 7 (P-frame). In this section, we describe a storage reorganization and access scheme that is adapted to MPEG stream dependencies and time-varying characteristics. The new mechanism provides a selected access mechanism that improves disk performance and minimizes the probability of frame loss.

3.1 Storage Pattern

To improve disk performance and accommodate the MPEG decoding algorithm, we defined a fixed T_{period} that is a multiple of the MPEG encoding group size. For example, with a group size of 10 (Fig. 1) and a video source that generates 30 frame/s, T_{period} is a multiple of 10/30 seconds. If the predefined period is T_{period}, the file system will read $30 \times T_{period}$ frames or $3 \times T_{period}$ MPEG patterns per session. Once the length of the period is fixed, we can define the number of frames (or MPEG groups) to be read every period. We reorganize the MPEG frame sequence as shown in Fig. 4. In this example, we assume the fixed length period T_{period} to be five seconds. Thus, the storage pattern contains 150 MPEG frames (15 I-frames, 45 P-frames and 90 B-frames). In the reorganized storage pattern, the data

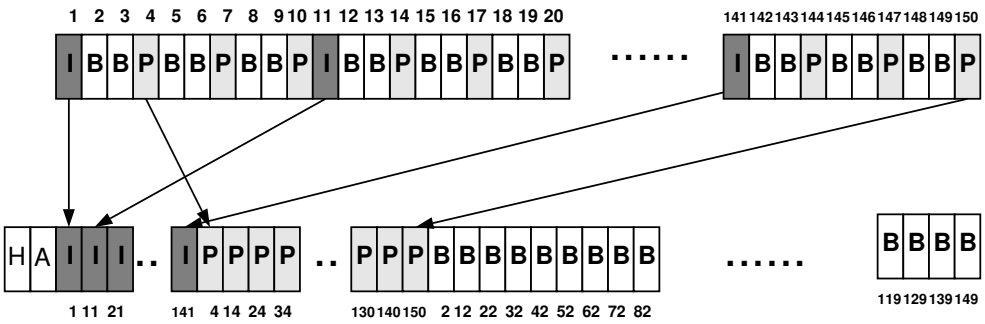

Figure 4: Storage Pattern

are placed contiguously in the order of importance (header, audio, I-frames P-frames then B-frames). **H** denotes a header containing information about the antecedent and succedent storage patterns (size of each frame) and **A** denotes the audio data. Each reorganized pattern is stored contiguously on the disk to reduce retrieval latencies. This storage organization accompanied by our refined round-robin scheduling scheme can help the file system predict the retrieval status for subsequent working periods. This allows the scheduler to adapt quickly to any changes in scheduling due to user input and to provide the best service.

3.2 Selected Access Scheme

The working period T_{period} in Eq. 1 must be greater than the time necessary to transfer data for all sessions from the disk to prevent starvation. However, as described in Section 2 and Fig. 2, the traffic rate for a MPEG-compressed video is not constant. Consequently, the traffic rate for the sum of all sessions $\sum_1^n S(i)$ on the disk is also a random variable. The file server accepts a set of sessions if their statistical sum is less than the preset period T_{period}. This means that several time varying MPEG video streams are statistically multiplexed during the disk retrieval phase. If p is the maximum starvation frequency that can be tolerated, we have

$$Prob[(\sum_{1}^{n} \frac{S(i)}{R} + \sum_{1}^{n} T_{latency}(i)) > T_{period}] < p \tag{2}$$

This means that the probability that the preset period T_{period} is less than the time to transfer data from disk is less than p. In Eq. 2, $T_{latency}(i)$ represents the seek latency when the disk switches service from session $i-1$ to session i. Because the data of subsequent

session i can be placed anywhere on the disk, the latency $T_{latency}$ is also a random variable. If $E(T_{latency})$ is the average seek latency, $\sigma^2_{latency}$ the variance of seek latency, $E(S)$ the average size of storage pattern, and σ^2_S the variance of the size of storage pattern, then the time to transfer data from the disk $T_{transfer}$ can be estimated as [3]:

$$T_{transfer} = \sum_{1}^{n} \frac{S(i)}{R} + \sum_{1}^{n} T_{latency}(i) \tag{3}$$

$T_{transfer}$ is a sum of random variables. By the Central Limit Theorem, if the number of sessions n is large enough, we know that the distribution of $T_{transfer}$ can be approximated by a Normal distribution. We can plot $T_{transfer}$'s probability density (Fig. 5) with a mean value $E(T_{transfer})$ and standard deviation $\sigma(T_{transfer})$. This means if the fixed length working period T_{period} and the number of session are given, we can predict the frequency of starvation p as shown in Fig. 5.

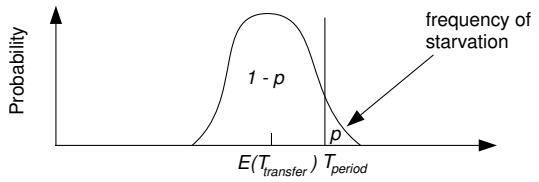

Figure 5: Distribution of T

As described in Section 3.1, we measure the size of every frame, encapsulate this information in a header and place it in the storage pattern for the previous group. With the reorganized storage layout, whenever a period is complete the file system has sufficient information about the requirements for the next period. The file system can then calculate the total size of the data that needs to be read for the next period and can predict the total

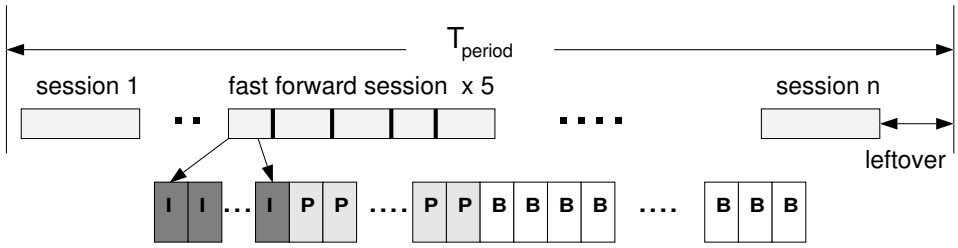

Figure 6: Access Scheduling for Fast-Forward

retrieval time. If starvation will occur in the coming period, it is known beforehand and the system can make the necessary adjustments. We now describe a "selected access scheme" to transfer the important data whenever starvation occurs.

We first calculate the time for the file system to transfer data from a disk by using Eq. 3. If the transfer time $T_{transfer}$ is less than the fixed-length working period T_{period} then there is no starvation for the next period. This means that the file system has sufficient time or bandwidth to retrieve data for all existing sessions. In this situation the file server scheduling process can ask the file system to read $S(i)$ of data for session i. $S(i)$ is the size of the data for session i in this period and the characteristics of $S(i)$ are already known by the scheduler at the end of previous period. If the transfer time $T_{transfer}$ is greater than the working period T_{period}, we can predict a starvation for the coming period. We then calculate the shortage time for the file system when it is unable to transfer data from the disk. We define the shortage time T_{fail} as

$$T_{fail} = T_{transfer} - T_{period} \tag{4}$$

If the amount of data the file system fails to read is S_{fail}, we have $S_{fail} = T_{fail}/R$. From Eqs. 3 and 4, we have

$$S_{fail} = \sum_{1}^{n} S(i) + R \sum_{1}^{n} T_{latency}(i) - R \times T_{period} \tag{5}$$

We can then reschedule the access scheme by selecting the frames to be dropped from existing sessions. Let m out of n existing sessions be selected to absorb the starvation. As a result, each selected session is only allowed to read $S(i) - S_{fail}/m$ amount of data. This means that every selected session drops S_{fail}/m of data and shares the impact of starvation. Because we reorganize the storage pattern on the disk, the file system will drop less important data predictably. As shown in Fig. 4, less important data are placed at the end of each storage pattern. Therefore, only B-frames will be dropped and the dropped B-frames will be uniformly distributed across the entire retrieved block. This can guarantee that no reference frames will be dropped thereby improving frame loss characteristics.

4 VCR-Like Functionality

In this section we describe how this new model can provide VCR features such as *reverse, fast-forward* and *fast-reverse*.[3]

Fast-forward can be interpreted as the display of more video content than normal playback in a given time interval. Let us assume that the fixed length period T_{period} is 5 seconds. If a user requests a fast-forward at five times the normal speed, the file server must read frames from parts of 5 storage patterns during a period. However since the client can play back video at 30 frame/s, only 150 frames from these 5 patterns can be played out during the period. This can be achieved by sending the 75 I frames from the 5 patterns and the remaining 75 as P and B frames, distributed equally between the 5 patterns.[4]

For block retrieval, the file server will read the entire contiguous block, filter out any unnecessary frames, and send only desired frames to the client. In this example, the file server must retrieve 750 frames (@5 × 5 s × 30 frame/s) during the period (5 s) or 150 frame/s (@5 × 30 frame/s) from storage. This means that the fast-forward session consumes 5 times the disk I/O bandwidth as compared to regular playback. This scheme is therefore inefficient and infeasible for a VOD server supporting multiple sessions. Let us consider the frame retrieval approach without our storage reorganization model. The server can prevent the retrieval of unnecessary frames; however, due to the frequent disk-head movement required to search for the desired video frames, this approach is still impractical. Using the assumptions from the previous example, if we read 15 I frames a second for a fast-forward session, there are 75 disk-head jumps during a period (5 s). If each disk-head jump takes 30 ms, the server will waste 2.25 s or 45% of the available bandwidth (in this case, period length $T_{period} = 5$ s; $2.25/5 = 45\%$).

Because our model reorganizes the MPEG frames in the physical storage device (Fig. 4), the disk scheduler has sufficient information about the future retrieval of frames (i.e., the server knows the size of each frame in advance). The server can dynamically retrieve desired frames in a sequential manner as shown in Fig. 6. When the user initiates a fast-forward, the scheme prevents the retrieval of any unnecessary frames and reduces the frequency of disk-head jumps. For a fast-forward at five times the regular speed, there are only 5 jumps (four more than regular playback) that take an additional 0.12s of disk latency.

[3]Functions such as *slow-play* and *pause* require lower bandwidth than regular playout and can be easily implemented.

[4]Because I frames are larger than P and B frames, there is an increase in the bandwidth requirement for this session during the fast-forward period. This issue is further discussed in the next section.

Reverse playback can be interpreted as displaying video frames in the reverse order. When the user requests reversal of video, the client station must request the server to retrieve the stored data in a reverse order. As described in Section 2, MPEG frames cannot be decoded independently. The server must send the MPEG encoding group by group in a reverse fashion to the client station (Fig. 7). When individual MPEG groups arrive at the client station, the MPEG decoder can decode the entire MPEG group and display the decoded frames in a reverse order.

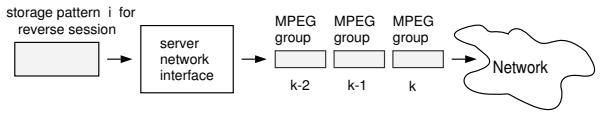

Figure 7: Data Flow for a Reverse Session

Fast-reverse is a scenario in which the video is played in reverse at high speed. When the user requests a fast-reverse, the server must reschedule the disk to access the desired frames in a reverse order as shown in Fig. 8. Here, we illustrate a 5 $\times$ fast-reverse session with a frame rate of 15 frame/s. The server needs to read all I frames from the 5 groups. Each I frame can be decoded independently and is sent as an individual group. If we scale up the frame rate to 30 frame/s for this fast-reverse session, the server must retrieve another 75 P frames in a period. In this case, we can treat one I frame and one P frame as an MPEG group and send them in a reverse order as described earlier.

5 Dynamic Resource Allocation and Admission Control

Thus far, we have described the storage and retrieval mechanisms to support multiple sessions with VCR-like features from a video server. Our approach is to allocate the available disk bandwidth among the different sessions such that the probability of starvation is limited (Eq. 2). The admission control policy can be stated simply as: "Allow a new session if the probability of starvation after allocating resources for it is less than a very small constant p." In this section, we elaborate on the dynamic nature of resource allocation in our scheme. A dynamic allocation mechanism has the following advantages over a static one, where resources allocated at the start of a session remain fixed during its lifetime:

Dynamic QOS: This feature allows the user to change the QOS provided by the application

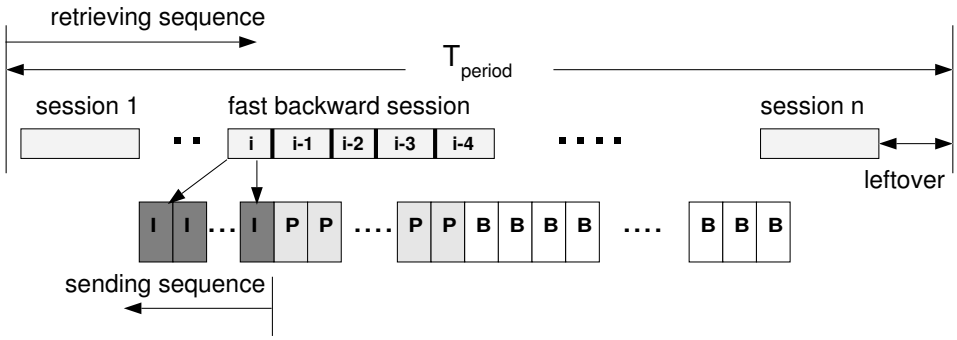

Figure 8: Access Scheduling for Fast-Reverse

while it is in progress. For example, if the user requests a change in the frame rate from 30 frames per second to 20, the bandwidth allocated for this session can be reduced to support the lower frame rate. In a static scheme, the user would still be allocated bandwidth corresponding to a higher frame rate. Though the unused bandwidth will be used by other sessions because the sessions are multiplexed, the admission control mechanism will not consider this bandwidth when a new session must be admitted. The user can also request an increase in frame rate, and be allocated the extra bandwidth if it is available. However, the request will be refused if sufficient bandwidth to support the new frame rate is not available.

Application of scaling gains: Due to limitations of human perception and tolerance for a limited degradation in quality, many video sessions are scalable. Thus, for a particular session, a user may be willing to tolerate a lower frame rate if they are charged a lower price. This allows the server to scale down some sessions in the event of congestion, or to support a new high priority session request.

Support for VCR-like features: As described in the previous section, a session might need extra bandwidth to support fast-forward. Because I and P frames typically require a larger than average bandwidth, it is very difficult to deliver 30 frame/s to the client station without degrading the existing sessions from the disk. Even if we only deliver 15 frame/s for this fast-forward session (send only 75 I frames), due to the low compression ratio (I frames) and the additional seek latencies (four more jumps) among the storage patterns, the bandwidth consumption is still higher than for regular playback. Our approach is to provide this additional bandwidth by scaling some of the other sessions which can tolerate a degradation in quality.

Our scheme supports the dynamic allocation scheme by reading only the most important

frames allowed by the bandwidth allocation for a session. The extra bandwidth available can be used to serve a new session, or improve the QOS of other existing sessions. Furthermore, our scheme allows a session to obtain the extra bandwidth needed for fast-forward or to support a new session by scaling down some of the existing sessions.

During connection set up, along with its bandwidth requirements, each session must also specify its threshold bandwidth, which is the bandwidth required to support the lowest QOS that can be tolerated by the user, and the "worth" of a connection which is a reflection of how much the user is willing to pay for the connection [10]. The admission control algorithm now reads: "Admit a session if the probability of starvation is less than p when all the sessions have been scaled." Thus, to generate the extra bandwidth needed to support a fast-forward request for a session, the server will scale down all lower priority sessions. If enough bandwidth to send the 150 frames (75 I and 75 P) during the 5 second period is still not available, the fast-forward will be scaled down to use only the available bandwidth. Thus, only 50 I frames may be sent, which will be played out at 10 frame/s.

The selected access scheme allows the server to take advantage of session scalability by accessing only the required information in every period to support the scaled sessions, thereby freeing resources to support other sessions. This is not possible when using a block access mechanism in which all data must be read by the server before it can scale a session. The frame access mechanism allows the server to select the frames to be read, but at the expense of increased seek overheads, since the required frames may be stored non contiguously on the disk.

6 Related Work

Several recent studies have analyzed the performance of disk retrieval schemes for continuous media and multimedia-on-demand. Rangan et al. [11] have developed an admission control algorithm for determining when a new concurrent access request can be accepted without violating the real-time constraints of existing sessions. Gemmell and Christodoulakis [7] have established some basic principles for retrieval and storage of delay-sensitive multimedia data. Keeton and Katz [9] have developed a placement scheme for multi-resolution video on disk arrays for providing scalable services. Chang and Zakhor [1] have proposed a placement strategy and an admission control strategy that provides scalable MPEG video service from disk arrays.

More recently, Jayanta et al. [8] have proposed a fast-forward and fast-reverse mechanism with an associated statistical quality-of-service guarantees. Chen et al. [5] have proposed a segment selection scheme to support variable fast-forward speed for MPEG video streams by placing independent MPEG segments in a disk array. In comparison, our scheduling and storage policies reorganize MPEG frames on a single disk. We make the possibility of missed deadlines known ahead of time to the scheduler using additional header information. This allows the scheduler to dynamically access desired frames.

7 Conclusion

In this paper we have proposed an MPEG frame storage reorganization technique to complement our selected access scheme. With the proposed placement scheme, MPEG-compressed data are prioritized and placed contiguously for retrieval from a storage device. The placement policy enables the retrieval process to guarantee the delivery of more important frames during periods of disk overload, thereby improving the quality of video playout in the system. We also show the ability of this scheme to support VCR-like functions without blocking the storage device by scaling the service of existing sessions. Simulation results show that the model yields improved throughput, increases reliability, and decreases the buffering needs [2]. These results are coupled with the proposed VCR-like control scheme to demonstrate that a single storage device can support several concurrent, fully-interactive multimedia sessions.

References

[1] Chang, E. and A. Zakhor, "Variable Bit Rate MPEG Video Storage on Parallel Disk Arrays," *Proc. IEEE 1st Intl. Workshop on Community Networking,* San Francisco, CA, July 1994, pp. 127-137.

[2] Chen, H.J., D. Venkatesh, and T.D.C. Little, "A Storage and Retrieval Technique for the Provision of Scalable MPEG Video," to appear in the *Journal of Parallel and Distributed Computing*, 1995.

[3] Chen, H.J., and T.D.C. Little, "Storage Allocation Policies for Time-Dependent Multimedia Data," to appear in *IEEE Trans. on Knowledge and Data Engineering,* 1995.

[4] Chen, M.S., D.D. Kandlur, and P.S. Yu, "Optimization of Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams," *Proc. ACM Intl. Conf. on Multimedia,* Anaheim, CA, August 1993, pp. 235-242.

[5] Chen, M.S., D.D. Kandlur, and P. S. Yu, "Support for Fully Interactive Playout in a Disk-Array-Based Video Server," *Proc. ACM Intl. Conf. on Multimedia,* San Francisco, CA, October 1994, pp. 391-398.

[6] Cidon, I., A. Khamisty, and M. Sidi, "Analysis of Packet Loss Processes in High-Speed Networks," *IEEE Trans. on Information Theory,* Vol. 39, No. 1, January 1993, pp. 98-108.

[7] Gemmell, J., and S. Christodoulakis, "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval," *ACM Trans. on Information Systems,* Vol. 10, No. 1, January 1992, pp. 51-90.

[8] Jayanta, K.D., J.D. Salehi, J.F. Kurose, and D. Towsley, "Providing VCR Capabilities in Large-Scale Video Server," *Proc. ACM Intl. Conf. on Multimedia,* San Francisco, CA, October 1994, pp. 25-32.

[9] Keeton, K., and R.H. Katz, "The Evaluation of Video Layout Strategies on a High-Bandwidth File Server," *Proc. 4th Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video*, Lancaster, U.K., November 1993, pp. 237-248.

[10] Krishnamurthy, A., and T.D.C. Little, "Connection Oriented Service Renegotiation for Scalable Video Delivery," *Proc. IEEE Intl. Conf. on Multimedia Computing and Systems*, Boston, MA, May, 1994, pp. 502-507.

[11] Rangan, P.V., H.M. Vin, and S. Ramanathan, "Designing an On-Demand Multimedia Service," *IEEE Communications Magazine,* Vol. 30, No. 7, July 1992 pp. 56-64.

[12] Sen, P., M. Maglaris, N.E. Rikli, and D. Anastassiou, "Models of Packet Switching of Variable-Bit-Rate Video Source," *IEEE Journal on Selected Areas in Communication,* Vol. 7, No. 5, 1989, pp. 865-869.