J. Indian Inst. Sci., Sept.–Oct. 2004, **84**, 123–140

Disk-oriented VCR operations for a multiuser VOD system

P. VENKATARAM*, SHASHIKANT CHAUDHARI**, R. RAJAVELSAMY, T. R. RAMAMOHAN** AND H. RAMAKRISHNA**
Protocol Engineering and Technology Unit, Electrical Communication Engineering Department, Indian Institute of Science, Bangalore 560 012, India.
**Central Research Laboratory, Bharat Electronics Ltd., Bangalore 560 013, Phone: +91-80-2838 1125.
e-mails: pallapa@ece.iisc.ernet.in, rajvel@protocol.ece.iisc.ernet.in, {shashi, rmohan, hr}@crlbel.ernet.in,
Phone: +91-80-2293 2747.

Received on July 02, 2003; Revised on March 16, 2004.

Abstract

Future high-speed computer networks are projected to carry dizzying array of networked multimedia applications including digital libraries, video and image servers, distance learning collaboration, networked virtual environments, and entertainment. A common threat to these applications is the so-called continuous nature of the data. Video-on-demand (VOD) application is a networked multimedia application where the subscribers access a remote server for the multimedia data preferably with VCR operations. These operations impose additional resource requirement on the VOD server in terms of storage space, retrieval throughput and network resources like bandwidth and buffers. In this work, we propose a cost-effective disk-oriented multi-user play-out system with VCR operations. The system uses segmentation of multimedia data into small segments and provides segment-skip mechanism for VCR operations and implementation.

Keywords: VOD, VCR operation, MPEG, switchover, time, video.

1. Introduction

Video-on-demand (VOD) is a service where subscribers access a remote VOD server. The server stores a large number of digital compressed multimedia streams, which can be transmitted to a subscriber on request. The multimedia streams consist of compressed video and audio, where the most popular standard is MPEG. The server is designed to act as a remote VCR player for each client. With recent advances in hardware, software, compression technology, high-bandwidth storage devices and high-speed networks, it is now possible to provide real-time VOD service over the Internet. There are two modes of providing VOD service over the Internet: download mode and streaming mode [1].

In the download mode, a user downloads the entire video file and then plays back. Here download time is large and it is unacceptable to the end user. In the streaming mode, the video file need not be downloaded in full, but is played out while parts of it are received and decoded. With the establishment of MPEG standards and increasing popularity of Internet, it is important to investigate how to implement efficiently an MPEG streaming system over the Internet. Video Cassette Recording (VCR) functionality such as fast forward (FF), fast backward (FB), rewind and pause are highly desirable in streaming VOD application.

*Author for correspondence.

In this paper, we propose an algorithm to introduce streaming mode VCR operations over the secondary storage in a multi-user VOD application.

1.1. *Problems involved in multimedia VOD application*

A VOD streaming system over Internet should be capable of delivering concurrent video streams to a large number of users. The realization of such a system presents several challenges, such as high storage capacity and throughput in a video server and high bandwidth in the network to deliver a large number of streams.

Large data transfer rate and storage space requirement: Digital video and audio playback consumes data at a very high rate. Thus, a multimedia server must provide efficient mechanisms for storing, retrieving and manipulating data in large quantities at high speeds.

Real-time storage and retrieval: Continuous media (such as audio and video) consist of sequence of media quanta (such as video frames or audio samples) which convey meaning only when presented continuously in time. This is in contrast to a textual object, for which spatial continuity is sufficient [2].

Supporting several concurrent interactive sessions: It is desirable for a VOD system to support several concurrent interactive sessions (supporting from the same physical device, i.e. hard disk). It becomes difficult for a VOD server to meet session timing requirements due to unpredictable nature of disk seek latencies [3]. Due to time-varying characteristics of compressed video, it is difficult to predict disk production and display consumption rates. Without proper scheduling, VOD system cannot support continuous playback for an unlimited number of sessions.

Protocols for streaming video: The transport protocols on top of IP, i.e. transport control protocol (TCP) and user datagram protocol (UDP) are not sufficient to run real-time applications such as VOD. They do not support synchronization within a single stream and also between two or more streams. Real-time protocol (RTP) and its associated real-time control protocol (RTCP) support real-time data transfer [4].

1.2. *Complexity of executing VCR operations on disk-based VOD system*

VCR operations impose additional resource requirement on the VOD server in terms of storage space, retrieval throughput and network bandwidth [5]. Moreover, video compression techniques such as MPEG impose additional constraints on the process since they introduce inter-frame dependencies.

The difficult challenge in design of the MPEG standard was the following: on one hand, the quality requirements demand a very high compression not achievable with intraframe coding alone; on the other hand, the random access requirements (such as FF, FB, rewind, pause, etc.) are best satisfied with pure intraframe coding [6]. To achieve balance between intra- and inter-frame coding, the members of MPEG standard have resorted to using two inter-frame coding techniques: predictive (P frames) and interpolative (B frames). The Intra frames (I frames) are intended to assist random access. With the I-B-P structure of frames within MPEG video, to decode a P frame, previously decoded I/P frame need to be decoded

first. To decode a B frame, both I/P frames before and after this B frame need to be decoded first. The inter-frame dependency introduced by P and B frames implies that it is not possible to play out every *n*th frame from the MPEG stream to achieve an FF speed of *n*.

It is also possible to play out only I frames from MPEG stream to achieve FF operation. But the frame rate has to be reduced significantly in order to maintain a fixed network bandwidth, because I frames are very large in size as compared to P and B frames. Therefore, the end subscriber will have to reduce playout rate during FF.

Thus, the challenge in providing VCR operations is to minimize storage space overheads, to minimize the increase in bit rate during FF operation and to provide acceptable video quality.

1.3. *Some of the existing works*

Some recent works have addressed the implementations of VCR operations in VOD applications using MPEG compressed streams. Dey-Sinclair *et al.* [7] have proposed an increased playback rate method. In this method, to achieve an FF speed of *n*, the playback rate (frames/s) should be increased by a factor of *n*. There are many disadvantages in this scheme. First, the decoder must be able to display video using increased playback rate. Secondly, it increases resource load on both server and network. For a speed-up of *n*, the server must retrieve *n* times as much data from a storage system and send *n* times as much data over a network. Also it is very complex to achieve FB operation.

Chen *et al.* [5] have proposed a segment skipping method. Here each segment (group of pictures, i.e. GOP) consists of consecutive frames beginning with an I frame and ending before another I frame. In this method, entire segments (GOPs) are either skipped or sent to the client. For example, to achieve a speed-up factor of *n*, every *n*th segment is sent to a client. The client continues to decode at a normal decoding rate. By skipping various number of segments, multiple playback rates can be achieved. There are two types of GOPs, closed ones which are independently decodable and end with a P frame or an I frame, and open ones which require one or more B frames from the previous GOP in order to be fully decoded and which end with a B frame. In segment skipping method, if open GOP is selected for decoding and displaying, some processing is required at the client side to discard B frames, which do not have associated I/P frames. Each segment may contain a large number of frames (10–15), thus skipping segment results in noticeable discontinuities in the sequence of frames being displayed.

Ghandeharizadeh *et al*. [8] have proposed offline approach, i.e. it implements the fast forward and fast backward functionalities by maintaining separate, pre-processed versions of the normal speed clips. This method requires extra disk storage.

Kenchammana-Hosekote *et al.* [9] have proposed a method for providing VCR operations, which uses rate variation and sequence variation technique. In rate variation technique rate of data units (i.e. in a video stream, frame or group of frames) is changed. Since a video stream is a timed sequence of data units, it results in a speed-up or slowdown of the stream. In sequence variation technique, the order in which data units are flowing in a stream is changed.

Chen *et al*. [8] have proposed the method for supporting FF, where only I frames are sent to the client. This method proposes to reorganize the MPEG stream in the order of importance (I frames, P frames, then B frames). Since I frames are very large in size as compared to P and B frames, the frame rate has to be reduced significantly during FF/FB operation to maintain a fixed network bandwidth. The end subscriber will have to reduce frame rate during FF operation. Otherwise, if the same frame rate is maintained during FF/FB operation, it requires more network bandwidth as size of the I frame is much higher than that of B and P frames. If there are 15 frames in a GOP, playing out only I frames will result in a FF speed of $n = 15$, which results in a more loss of information during FF/FB operation. This method works well when a user is searching for a particular scene in the movie, where the speed of the actors is normal. But when the user is searching for a particular scene in a movie where the speed is very high, e.g. fast gymnastics motion in the Olympic clip, it is not acceptable since it results in more loss of information.

The two disadvantages, i.e. increase in network bandwidth, and loss of information for a clip where actors are moving faster, are removed in our proposed method which uses skipping sub-segment method rather than skipping all B and P frames.

1.4. *The proposed method*

The method proposed by us for execution of VCR operations on secondary storage-based VOD is by skipping sub-segments. The sub-segment is an independent sequence in a segment, which can be decoded independently. Segment start offset and sub-segment end offset are stored in segment offset table to help in retrieving segment and sub-segment, respectively. The segment and sub-segment form a basic unit of retrieval during normal and FF playout, respectively. On issuing of FF command, sub-segments are retrieved with the aid of segment offset table and delivered to a client.

1.5. *Organization of the paper*

The remainder of the paper is organized as follows: In Section 2, the general issues involved with storage structure, retrieval techniques, and playout method in relation to VOD are described. Section 3 describes the storage structure, logical segment organization, and execution of VCR operations in relation to our proposed playout method. In Section 4, we have presented an analytical model of a VOD server to find out the maximum time taken for a switchover from normal to FF playout and the maximum time taken for a switchover from FF to normal playout. Description of the proposed VOD system with VCR operations is described in Section 5. Results are given in Section 6. The paper concludes with Section 7.

2. VOD on secondary storage

In a VOD server storage hierarchy, RAM, which is the primary storage, is at the top of the pyramid. It is the fastest storage medium, but also the most expensive. The second in the pyramid is the secondary storage, i.e. magnetic disk. It has short access time, high transfer rates, and substantial capacities, which makes them well suited for VOD server application.

There are several challenging issues relating to designing storage system for VOD application, such as high throughput, large capacity and fault tolerance. The VOD server must provide efficient methods for storing and retrieving large amounts of data at high speeds. If the entire video file is stored on one disk, the number of concurrent accesses to that file is limited by the throughput (disk I/O bandwidth) of the disk. To overcome this limitation, data stripping [11] was proposed. Under data stripping scheme, a multimedia file is scattered across multiple disks and the disk arrays can be accessed in parallel.

There are two methods of retrieval of data from multimedia server: server-push and client-pull [12]. In the server-push system, a client makes an initial request to the server to start delivery of data. The server continues to deliver data until the end of the stream is reached or the client requests some type of VCR operation. The server has complete control of how and when accesses to the storage media are executed. If a client wants an FF operation, he must send a message to the server which takes an appropriate action to the operation. In the client-pull system, the server retrieves data for a client only in response to an explicit read request. A client must ensure continuous playback by issuing periodic read requests. Thus the client-pull system is inherently stateless, and the server-push system must maintain client state. The server-push system is more appropriate for VOD system, while client-pull system is more suitable for accessing textual/numeric data.

The playout system at the server side has to continuously transmit media parts to the clients. Also, the server has to maintain the VCR operation status of various clients and service their requests. The playout system at the client side has to receive and display continuously the media parts transmitted by the VOD server. Also it has to send the VCR operations request to the VOD server.

3. Proposed VOD playout system

The objective of our system is to propose a cost-effective single disk-based streaming mode, server-push type VOD server that can support VCR operations for multiple users. The system uses skipping sub-segments method for VCR operations. The sub-segment called the Fast Forward Segment is an independent sequence in a segment which can be decoded independently. We refer the sub-segment as FF segment. Segment start offset and sub-segment end offset are stored in segment offset table to help retrieve segment and sub-segment, respectively. The segment and sub-segment form a basic unit of retrieval for normal and FF playout, respectively. In the streaming mode, the video content need not be downloaded in full, but can be played out while parts of the contents are being downloaded. The important challenge is how to divide the media stream into parts. If the compression scheme used does not introduce inter-frame dependencies into the compressed video, video stream can simply be divided into segments (i.e. parts) according to the desired segment size. Therefore, some additional provision is needed to divide video stream in parts, as inter-frame dependencies exist in MPEG stream [5].

Consider an MPEG video stream consisting of I, P, and B frames as shown in Fig. 1. In this stream, I frames are coded such that they are independent of any other frames. P frames are coded such that they have dependency on the preceding I/P frame. Similarly, B frames are coded such that they depend on two anchor frames: the preceding I/P frame and the

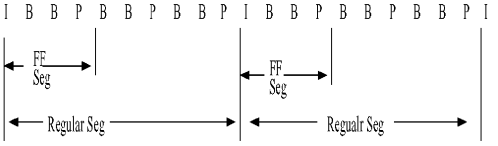

FIG. 1. A logical organization of segments.

Segment number	Regular segment start offset (bytes)	FF segment end offset (bytes)
0	0	28700
1	70100	100100
2	139225	169325
3	209325	239125
.		
.		
m		

FIG. 2. Segment offset table.

following I/P frame. The inter-frame dependency implies that it is not possible to decode a P frame without the preceding I/P frame. Similarly, it is not possible to decode B frame without the corresponding I and/or P frame. To comply with the inter-frame dependency, a multimedia stream is logically divided into two segments: regular and FF.

Regular segment: Each regular segment consists of frames beginning with an I frame and ending before another I frame. Assume that in MPEG compressed video, there are $2k + 1$ BBP frames between any two consecutive I frames, where k is a positive integer. There are $2k$ BBP sequences between any two consecutive IBBP sequences in the video. As shown in Fig. 1, the order of frames within a video stream is I B B P B B P B B P, for which $k = 1$. Thus, there are three BBP sequences between any two consecutive I frames and two BBP sequences between two consecutive IBBP sequences of frames. Here, for $k = 1$, the regular segment consists of I B B P B B P B B P frames. The regular segment forms a primary unit of retrieval during normal playout.

FF segment (*sub-segment*)*:* Each FF segment consists of frames beginning with an I frame and ending after the first P frame in the sequence. Thus, FF segment is obtained by omitting the $2k$ BBP sequences between two consecutive IBBP sequences in the video. As shown in Fig. 1, the first four frames, i.e. I B B P, form the FF segment in our method. The FF segment is considered independent sequence because the frames within this segment can be decoded independently. The FF segment forms a primary unit of retrieval during FF/FB playout.

The total video, which is in MPEG format, is divided into finite number of regular segments. These segments are scattered on the disk system. We also maintain a segment offset table in the RAM (Fig. 2). Initially, regular segment offset will be 0 for segment number 0. Then the whole MPEG video file is parsed. Whenever the start of regular segment and end of FF segment are found, its offset (in terms of bytes) is stored in the segment offset table for a corresponding segment number.

3.1. *Execution of VCR operations*

Various types of VCR operations used in the literature are pause, rewind, stop, jump forward, jump backward, speed up, fast reverse, FF, FB, forward search, backward search, etc. Of these, the meaning of the terms like speed-up, FF, forward search are the same, i.e.

quickly moving presentation forward. Similarly, the meaning of the terms like fast reverse, FB, backward search are the same, i.e. quickly moving presentation backward. Jump forward/backward terms are used to jumping to a target time of presentation in the forward/backward direction. Pause means temporarily stopping the presentation. Similarly, rewind means playing a presentation from the start. In this paper, we are using the pause, FF, FB and rewind operations for providing VCR functionality.

FF operation: After receiving the FF request, the server reads FF segment data for the corresponding current segment number and transmits it to the client. For example, assume that a server is reading the 20th regular segment from disk and the request for the FF operation is received. In this case, after reading and transmitting the 20th segment to the corresponding client, the server reads and transmits the 21st FF segment, then the 22nd FF segment and so on until new VCR operation request is received or until the end of MPEG video file as shown in Fig. 3. In this case, four frames from one segment are sent during FF operation and the rest of the six frames are skipped. Thus the speed-up rate of 2.5 is achieved.

FB operation: For FB operation, FF segments are read from disk in the backward direction. For example, assume that a server is reading 20th regular segment from disk and the request for FB operation is received. In this case, after reading and transmitting 20th segment to the corresponding client, the server reads and transmits the 19th FF segment, then the 18th FF segment and so on until new VCR operation request is received or until the start of MPEG file as shown in Fig. 3. If there are four frames in the FF segment, all the four will be displayed in forward direction. As we are displaying these FF segments in reverse order (19, 18, 17, etc.), its effect is like fast backward operation and is still acceptable to the end user. In this case, four frames from one segment are sent during FB operation and the rest of the six frames are skipped. Thus the speed-up rate of 2.5 is achieved.

Rewind operation: For rewind operation, the server sets the next access segment to initial segment (segment 0). The server reads the regular segment zero and transmits it to the corresponding client. For example, assume that a server is reading 20th regular segment from

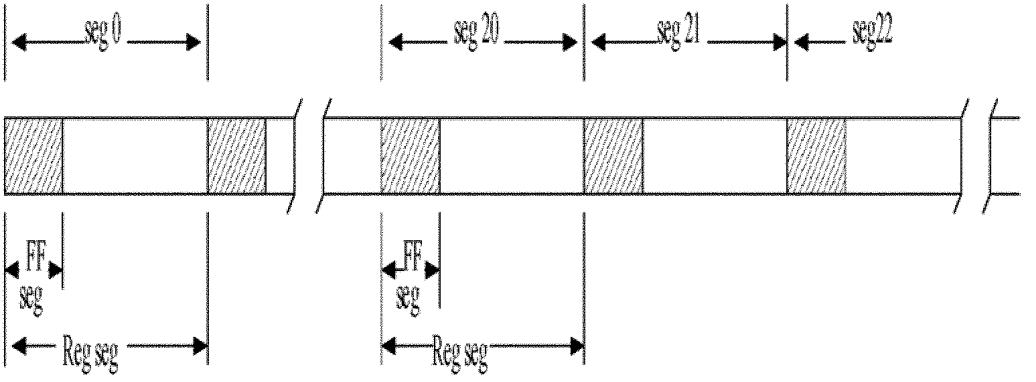

FIG. 3. Sub-segment skipping operation.

Client Id	Current segment number	VCR operation status
0	8	Normal
1	12	FF
2	4	FB
.		
.		
.		
n		

FIG. 4. Client status table.

disk and the request for rewind operation is received. In this case, after reading and transmitting the 20th segment to the corresponding client, the server reads and transmits the 0th segment and waits for new VCR operation request (Fig. 3).

Pause operation: For this operation, the server stops all operations until new VCR operation request is received. For example, assume that a server is reading the 20th regular segment from disk and the request for pause operation is received. In this case, after reading and transmitting the 20th segment to the corresponding client, the server will stop reading segment and transmitting it until new VCR operation request is received (Fig. 3).

3.2. *Retrieval of video data for a single client*

In order to retrieve the exact amount of data consisting of regular and FF segment, some indexing mechanism should be implemented at the server side. When a request from a client comes to start the movie, the VOD server continuously retrieves and transmits the data for *i*th segment, where the *i*th segment starts at *i*th regular segment start offset and ends at *i*+1th regular segment start offset minus one.

As shown in Fig. 2, the server initially reads bytes from offset 0 to offset 70100, i.e. 70099 bytes of data from disk corresponding to segment number 0 and transmit it. Next, the server reads bytes from offset 70100 to offset 139225, i.e. 69124 bytes of data from the disk corresponding to segment number 1 and transmit it, and continues reading every segment and transmitting it until *n*th segment is transmitted. The server maintains a status table in the RAM. It consists of status of the clients in terms of the requests of the clients (normal playout, FF playout, FB playout, etc.). The contents of the status table are client id, current segment number and VCR operation status for each client as shown in Fig. 4. The main algorithm for playout is given here. The server stores client status in the RAM and services the client requests until the end of MPEG video file.

3.3. *Retrieval of video data for multiple clients*

To retrieve data for multiple clients, we chose round-robin service scheme. Here, during each round one segment (regular/FF) is read from the disk and put into the buffer space of an active session. Thus video data is retrieved one segment at a time for an active session during each round.

Playout algorithm

```
begin
Store client requests in the RAM;
While (not end of MPEG file)
{
    If (VCR_operation_request = = normal_playout)
    {
        Read regular segment from disk;
        Increment current segment number stored in
        the RAM for corresponding client;
    }
    If (VCR_operation_request = = FF_playout)
    {
        Read FF segment from disk;
        Increment current segment number stored in
        the RAM for corresponding client;
    }
    If (VCR_operation_request = = FB_playout)
    {
        Read FF segment in backward direction from disk;
        Decrement current segment number stored in
        the RAM for corresponding client;
    }
    If (VCR_operation_request = = rewind)
    {
        Read regular segment starting from segment
        number 0 from disk;
    }
    If (VCR_operation_request = = pause)
    {
        Stop reading from disk;
    }
    Transmit regular/FF segment to corresponding client;
    If (new_VCR_operation_request)
    {
        Update VCR operation request in the RAM;
    }
}
end
```

4. Analytical model

In this section, we discuss an analytical model for a disk-oriented multiuser VOD system with VCR operations. We have derived the performance parameters for VOD system, such as switchover time from normal playout to FF playout, switchover time from FF playout to

normal playout, maximum number of concurrent client sessions, speed-up factor during FF and FB operation and average connection time of the user. We need to know the maximum switchover time to determine the amount of client buffer needed to guarantee video playback continuity.

Consider a VOD server that is servicing multiple clients for normal playout. Let the clients be $\{C_1, C_2, C_3 \ldots C_n\}$. Let the playout times for these clients be $\{P_1, P_2, P_3 \ldots P_n\}$, respectively, where $P_i \leq P_{i+1}$, $0 < i < n$. Let the number of regular segments be $\{S_1, S_2, S_3 \ldots S_m\}$ where $S_i = i$th regular segment.

Similarly, let the number of FF segments be $\{FS_1, FS_2, FS_3 \ldots FS_m\}$ where $FS_i = i$th FF segment. Let the size of the regular segments be $\{S_{s1}, S_{s2}, S_{s3} \ldots S_{sm}\}$, where S_{si} = size in bits of the ith regular segment.

Similarly, let the size of FF segments be $\{FS_{s1}, FS_{s2}, FS_{s3} \ldots FS_{sm}\}$, where FS_{si} = size in bits of the ith FF segment.

Assume that a server has a single disk with disk I/O bandwidth r_t. Also, assume that the rate of video be r_d. Similarly, assume that the network bandwidth be n_t.

The disk I/O bandwidth, r_t, is generally much higher than the rate of video, r_d. Based on this, the playout time in seconds for the ith regular segment, P_{si}, can be given as,

$$P_{si} = S_{si}/r_d. \tag{1}$$

Similarly, the playout time in seconds for the ith FF segment, PF_{si}, can be given as,

$$PF_{si} = FS_{si}/r_d. \tag{2}$$

Time taken to read regular segment from the disk, t_r, can be given as,

$$t_r = S_{si}/r_t. \tag{3}$$

Similarly, time taken to read FF segment from the disk, t_f, can be given as,

$$t_f = FS_{si}/r_t. \tag{4}$$

4.1. *Switchover time from normal to FF playout*

Switchover time from normal playout to FF playout can be defined as the time taken at the client side to switchover from normal playout to FF playout after the user has pressed the FF button. This switchover time should be minimal. Assume that the FF request comes at a time when the server has just started reading the data for the ith regular segment. The maximum time taken for switchover from normal playout to FF playout, $t_{\max 1}$ consists of two parts: delay at the server side, (t_s), and the network delay, (t_n).

$$t_{\max 1} = t_s + t_n. \tag{5}$$

The server side delay, t_s, is a sum of the time taken to read current regular segment from the disk after the FF command has been received, (t_r), time taken to start reading from the next FF segment, (T_{latency}), and time taken to read next the FF segment from disk (t_f).

$$t_s = t_r + T_{\text{latency}} + t_f. \tag{6}$$

Substituting eqns (3) and (4) in eqn (6), we have

$$t_s = S_{si}/r_t + T_{\text{latency}} + FS_{si}/r_t. \tag{7}$$

The network delay, t_n, is a sum of time taken to receive FF command at the server after the FF button is pressed at the client, (t_{n1}) and the time taken to send FF segment to the client (t_{n2}).

$$t_n = t_{n1} + t_{n2}. \tag{8}$$

As the FF command size is only 1 byte, we can neglect the actual time required to send FF command over the network. We will only consider delays involved at each hop while the command is being received at the server. Let t_h be the random variable which denotes the delay involved at each hop (queuing and processing delay at network nodes) and let b be the variance of the delay involved at each hop. Let y be the number of hops between the client and the VOD server. Queuing delay is variable and depends on the level of congestion and also it varies from packet to packet. Let a denote the average rate in packets/s at which packets arrive at the queue. Let R be the transmission rate in bits/s. Also suppose that all packets consist of L bits. Then the average rate at which the bits arrive at the queue is La bits/s. Assuming that the queue is very big, the evaluation of the queuing delay is decided by the ratio La/R, called the traffic intensity. If $La/R > 1$, then the queuing delay will tend to increase without bound. If La/R is close to zero, queuing delay will be close to zero. When La/R is close to 1, queuing delay increases rapidly. Processing delay is the time required to examine the packet's header and determine where to direct the packet. It also includes checksum calculation time and the transfer time from an input to an output port. On today's high-speed routers it is typically less than 30 μs.

Hops in a network are uniformly distributed in the interval $[x_1, x_2]$. Therefore, expected or the mean number of hops in the network is given by $E(y) = (x_1 + x_2)/2$. Let TF be the traffic factor which is a random variable whose value depends on the traffic at each node. $TF = 1$ indicates low load traffic and $TF = 2$, high load traffic. Now, the time taken to receive FF command at the server after FF button is pressed at the client, t_{n1}, can be calculated as,

$$t_{n1} = \sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b). \tag{9}$$

Similarly, the time taken to send FF segment to the client, t_{n2}, can be calculated as,

$$t_{n2} = FS_{si} / n_t + \sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b). \tag{10}$$

From eqns (5–10), $t_{\max 1}$ can be given as,

$$t_{\max 1} = S_{si}/r_t + T_{\text{latency}} + FS_{si}/r_t + FS_{si}/n_t + \sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b) + \sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b), \tag{11}$$

$$t_{\max 1} = (S_{si} + FS_{si})/r_t + T_{\text{latency}} + FS_{si}/n_t + 2\sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b). \quad (12)$$

The disk seek latency time, T_{latency}, can further be calculated as [13],

$$T_{\text{latency}} = T_{\text{cross}} + T_{\text{switch}} + T_{\text{rotate}}, \quad (13)$$

where T_{cross} is the arm positioning time required for the disk head to move to the current track, T_{switch} is the delay to switch the head to other surface, and T_{rotate} is the delay for disk rotation.

4.2. *Switchover time from FF to normal playout*

Switchover time from FF playout to normal playout can be defined as the time taken at the client side to switchover from FF to normal playout after the user has pressed the normal playout button. This switchover time should be minimum. Assume that the normal playout request comes at a time when the server has just started reading the data for the *i*th FF segment. The maximum time taken for switchover from FF playout to normal playout, $t_{\max 2}$ consists of two parts: delay at the server side (t_s), and the network delay (t_n).

$$t_{\max 2} = t_s + t_n. \quad (14)$$

The server side delay, t_s, is the sum of the time taken to read current FF segment from the disk after normal playout command has been received, (t_f), time taken to start reading from the next FF segment (T_{latency}), and the time taken to read next regular segment from the disk (t_r).

$$t_s = t_f + T_{\text{latency}} + t_r. \quad (15)$$

Substituting eqns (3) and (4) in eqn (15), we have

$$t_s = FS_{si}/r_t + T_{\text{latency}} + S_{si}/r_t. \quad (16)$$

The network delay, t_n, is a sum of time taken to receive normal playout command at the server after normal playout button is pressed at the client, (t_{n1}), and the time taken to send regular segment to the client (t_{n2}).

$$t_n = t_{n1} + t_{n2}. \quad (17)$$

As the normal playout command size is only 1 byte, we can neglect the actual time required to send the normal playout command over the network. We will consider only delays involved at each hop while the command is being received at the server. Let t_h be the random variable which denotes the delay involved at each hop (queuing and processing delay at network nodes) and let b be the variance of the delay involved at each hop. Let y be the number of hops between the client and the VOD server. The hops in a network are uniformly distributed in the interval $[x_1, x_2]$. Therefore, expected or the mean number of hops in the network is given by $E(y) = (x_1 + x_2)/2$. Let TF be the traffic factor which is a random variable whose value depends on traffic at each node. $TF = 1$ indicates low load traffic and $TF = 2$ high load traffic. Now, the time taken to receive normal playout command at the server after normal playout button is pressed at the client, t_{n1}, can be calculated as,

$$t_{n1} = \sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b). \tag{18}$$

Similarly, the time taken to send regular segment to the client, t_{n2}, can be calculated as,

$$t_{n2} = S_{si}/n_t + \sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b). \tag{19}$$

From eqns (14–19), $t_{\max 2}$ can be given as,

$$t_{\max 2} = FS_{si}/r_t + T_{\text{latency}} + S_{si}/r_t + S_{si}/n_t + \sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b) + \sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b), \tag{20}$$

$$t_{\max 2} = (S_{si} + FS_{si})/r_t + T_{\text{latency}} + S_{si}/n_t + 2\sum_{i=1}^{E(y)} (t_h + (-1)^{TF} * t_h * b). \tag{21}$$

The only difference between eqns (12) and (21) is the time required to send regular and FF segment over the network. As the size of the regular segment is more than the size of FF segment, we can conclude that $t_{\max 1} < t_{\max 2}$.

4.3. *Example*

For our prototype VOD system, we have used Seagate hard disk with the parameters as shown in Fig. 5. We consider the average size of regular segment for the MPEG movie (*The Good Son*) equal to 70 K bytes (S_{si} = 70 K bytes), and that of the FF segment 30 K bytes (FS_{si} = 30 Kbytes). Also we consider following parameters for the calculation of switchover time. Disk I/O bandwidth, r_t = 100 Mbytes/s, MPEG source rate, r_d = 1.5 Mbps, network bandwidth, n_t = 100 Mbps, delay involved at each hop, b = 0.1, expected number of hops, $E(y)$ = 5. Now the maximum switchover time from regular playout to FF playout, $t_{\max 1}$ = 0.11895 seconds (0.001 + 0.00555 + 0.0024 + 0.11). Similarly, the maximum switchover time from FF playout to regular playout, $t_{\max 2}$ = 0.12215 seconds (0.001 + 0.00555 + 0.0056 + 0.11).

Storage capacity	20 Gbytes
Average seek time	8.9 ms
Average latency	5.55 ms
No. of cylinders	16,383
Gaurenteed sectors	39,102,336
Bytes/sector	512
Sectors/track	63
Disk I/O bandwidth	100 Mbytes/s

FIG. 5. Parameters for Seagate hard disk, model no. ST370413A.

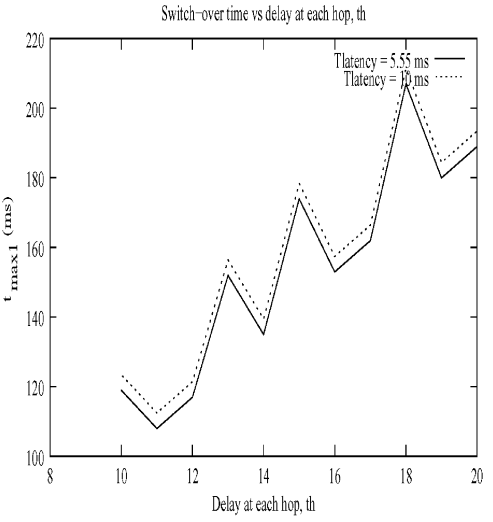

FIG. 6. t_{max1} vs delay at each hop, t_h.

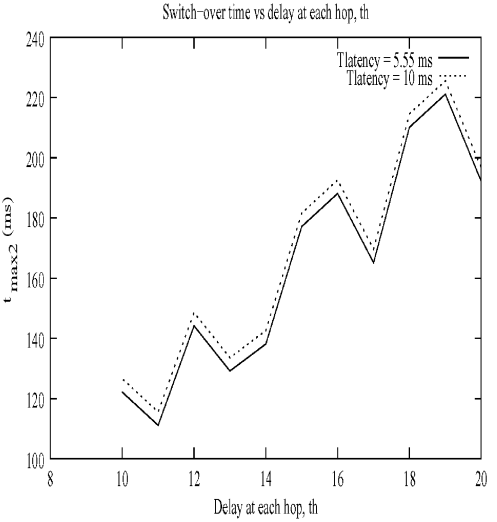

FIG. 7. t_{max2} vs delay at each hop, t_h.

To determine the traffic factor *TF*, which is required for calculating the values of delay at each hop, we have generated a random number between 1 and 100. If the random number is even, $TF = 2$, and if odd, $TF = 1$. *TF* is calculated 50 times, and finally the average value is taken. Figure 6 shows the variation of t_{max1} for various values of delays involved at each hop. Similarly, Fig. 7 shows the variation of t_{max2} for various values of delays involved at each hop. The variation in t_{max1} and t_{max2} is due to the fact that the traffic factor, *TF*, which depends on the actual traffic on the network, is changing randomly. From Figs 6 and 7, we conclude that switchover time from normal to FF playout, t_{max1} is less than that of FF to normal playout, t_{max2}. Similarly, we conclude that the switchover time increases as we increase $T_{latency}$.

4.4. *Maximum number of client sessions*

Maximum number of client sessions will be limited by disk I/O bandwidth, r_t, and the network bandwidth, n_t. Let p be the maximum number of concurrent client sessions that will be supported by the server. If $n_t \leq r_t$, the maximum number of concurrent client sessions will be,

$$p = n_t / r_d. \tag{22}$$

Similarly, if $n_t > r_t$, maximum number of concurrent client sessions [14] is given as,

$$p = r_t / r_d. \tag{23}$$

4.5. *Speed-up factor during FF and FB playout*

Assume that in MPEG compressed video, there are $2k + 1$ BBP frames between any two consecutive I frames, where k is a positive integer. Let n_r be the number of frames in a

regular segment, and n_f be the number of frames in an FF segment. Let S be the speed-up factor during FF and FB operations. Since the BBP sequence has three frames, n_r can be given as,

$$n_r = 3(2k + 1) + 1; \tag{24}$$

$$n_r = 2(3k + 2). \tag{25}$$

Similarly, n_f can be given as,

$$n_f = 3 + 1 = 4. \tag{26}$$

Speed-up factor, S, can be given as,

$$S = n_r/n_f. \tag{27}$$

Substituting eqns (25) and (26) in eqn (27), we have,

$$S = (3k + 2)/2. \tag{28}$$

4.6. *Average connection time*

Average connection time, T, of a user can be defined as the time from the moment the user is connected to a video server to the time when the user is disconnected [15]. Suppose the

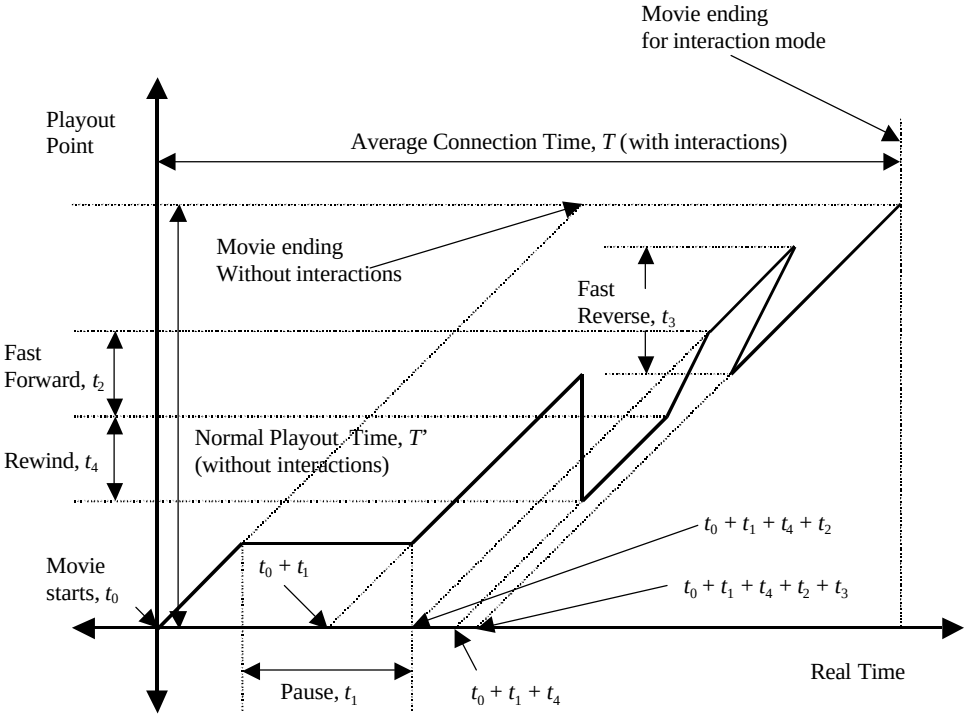

FIG. 8. Average connection time.

proportions for various interaction operations are as follows: stop/pause, q_1; FF, q_2; FB, q_3; rewind, q_4, where $\sum_{i=1}^{n} q_i = 1$. Let S be the speed-up factor for FF and FB operations. Consider the normal playout time of a video program without user interactions is T'. Now T may be longer or shorter than T' depending on the user interactions. For example, if we stop the video for t_1 time units, T will be increased by t_1. FF with a speed-up factor of S will decrease T by $t_2E(S-1)$, while FB will increase T by $t_3E(S+1)$. Similarly, rewind operation after t_4 units of time will increase T by t_4. $E(t_i)$ is the expected time of interactions of a user. Therefore, the average connection time, T, can be given as,

$$T = T' + q_1E(t_1) - q_2E(t_2)(S-1) + q_3E(t_3)(S+1) + q_4t_4. \tag{29}$$

The effect of pause, FF, FB, and rewind operations on the average connection time can be shown graphically (Fig. 8). To explain the effect, we define actual and logical start times. The actual time is defined as the time at which the movie starts playing. The logical start time is defined as the time minus the time it would have taken to watch the movie from its beginning until the point at which the customer was watching without any interruption. Initially, the logical and actual start times are the same, but become different when interactions are started. For example, Fig. 8 shows how the logical start time changes because of pause operation of duration t_1. When the normal playout is occurring, the slope of the line is equal to one. During pause operation, the slope becomes zero because the real time continues, but the playout point is not advancing. Similarly, FF operation causes a jump in an earlier logical start time.

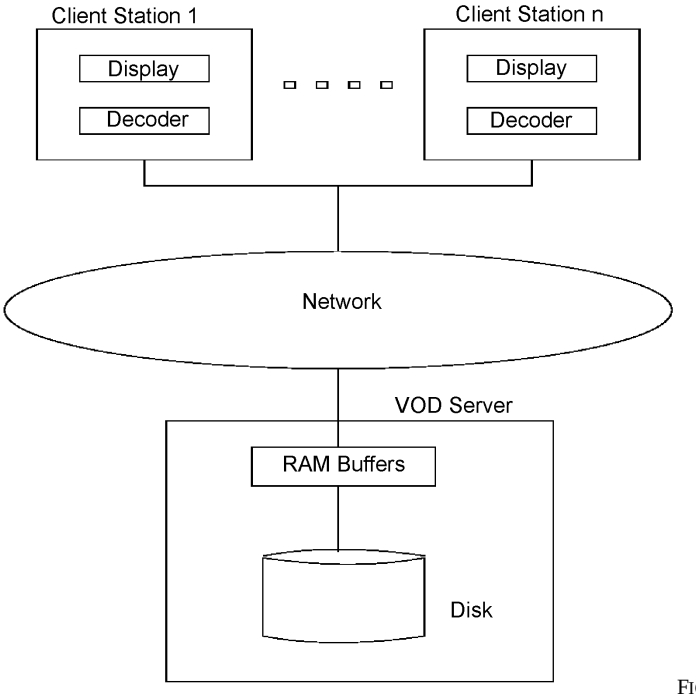

FIG. 9. A disk-based VOD server.

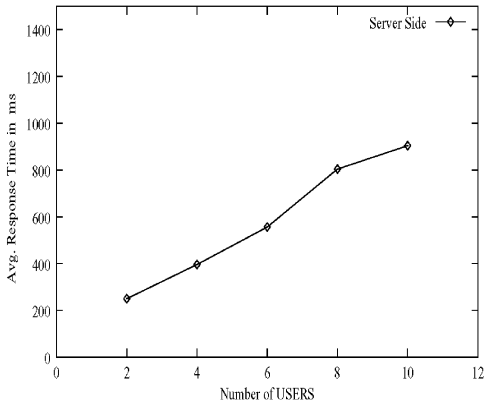

FIG. 10. Average response time at the server vs number of users.

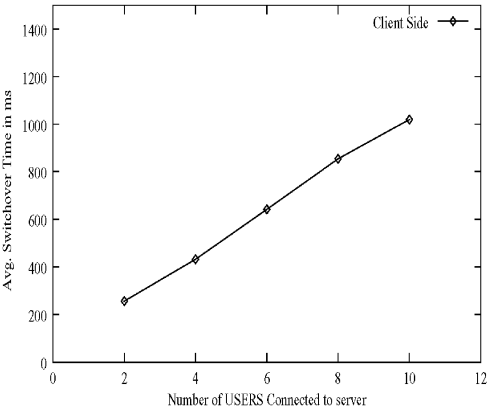

FIG. 11. Switchover time at the client vs number of users connected to server.

5. Prototype implementation

We have implemented the proposed VOD playout system on wire and wireless network available at the inhouse PET Unit at the Indian Institute of Science. The model for the proposed disk-based VOD system is shown in Fig. 9. We have used Seagate hard disk with the parameters shown in Fig. 5 for our prototype implementation. We have kept on the VOD server video data which runs for 100 min. The server maintains the segment offset table in the RAM, which stores the regular segment start offset and FF segment end offset. With the help of this status table, the server services the various random access requirements of multiple clients. For this video data we have 4834 segments. We have conducted the experiments with several sets of clients retrieving the data at different times. We discuss some of the results in the following section.

6. Results

Response time is defined as the time between generation of a request and the completion of the operation requested. Figure 10 shows that the response time of the server increases as the number of clients increases. Switchover time can be defined as the time taken at the client side to switchover from normal to FF playout or vice versa. Figure 11 shows that the switchover time increases as the number of clients increases at the server side. As more users are connected to the server, the response time and switchover time increases slowly, as it requires more resources at the server side.

7. Conclusions

In this paper, we have proposed a method of VCR operations on secondary storage-based VOD server. Algorithm for playout is proposed which supports various VCR operations such as FF, FB, rewind and pause. For FF and FB operations, sub-segment skipping method is used and is quite acceptable to end-user. During these operations, the frames which can be decoded independently from the regular segment are transmitted to the client and the rest

of the frames from the regular segment are skipped, thus achieving a speed-up factor. We have also described an analytical model for a disk-oriented multiuser VOD system with VCR operations. With the help of this analytical model, we have derived various performance parameters for VOD system, such as switchover time from normal to FF playout, switchover time from FF to normal playout, maximum number of concurrent client sessions, speed-up factor during FF and FB operations and average connection time of the user. Finally, we conclude that switchover time from normal to FF playout, $t_{\max 1}$, is less than the switchover time from FF to normal playout, $t_{\max 2}$.

Acknowledgment

This work is sponsored by the Central Research Laboratory, Bharat Electronics Ltd, Bangalore 560 013. We wish to thank Mr. Sunil Kumar Manvi for his comments and discussions.

References

1. D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang and J. M. Peha, Streaming video over internet: Approaches and directions, *IEEE Trans. Circuits Systems Video Technol.*, **11**, 282–300 (2001).
2. P. V. Rangan, H. M. Vin and S. Ramanathan, Designing an on-demand multimedia service, *IEEE Commun. Mag.*, **30**, 56–65 (1992).
3. H. J. Chen, T. D. C. Little and D. Venkatesh, A storage and retrieval technique for scalable delivery of MPEG encoded video, *J. Parallel Distributed Computing*, **30**, 180–189 (1995).
4. P. Liu, Internet protocols for multimedia communications, Part 1: Ipng–The foundation of internet protocols, *IEEE Multimedia*, **4**, 85–90 (1997).
5. 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, *ACM Multimedia Systems*, **3**, 126–135 (1995).
6. D. Le Gall, MPEG: A video compression standard for multimedia applications, *Commun. ACM*, **34**, 47–58 (1991).
7. J. Dey-Sinclair, J. Saleni, J. Kurose and D. Towsley, Providing VCR capabilities in large-scale video servers, *Proc. ACM Multimedia*, San Francisco, CA, pp. 25–32 (1994).
8. S. Ghandeharizadeh, R. Zimmermann, S. H. Kim, W. Shi and J. Al-Marri, Scalable video browsing techniques for intranet video servers, *Seventh Workshop on Information Technologies and Systems* (*WITS '97*), Atlanta, Georgia, December 13–14, 1997, pp. 210–218.
9. D. R. Kenchammana-Hosekote and J. Srivastava, I/O scheduling for digital continuous media, *Multimedia Systems*, **15**, 213–237 (1997).
10. H. Chen, A. Krishnamurthy, T. Little and D. Venkatesh, A scalable video-on-demand service for the provision of VCR-like functions, *Proc. IEEE Int. on Conf. Multimedia Computing Systems*, Washington DC, pp. 65–72 (1995).
11. P. Shenoy and H. M. Vin, Efficient stripping techniques for multimedia file servers, *Perform. Eval*., **38**, 175–199 (1999).
12. P. Shenoy, P. Goyal and H. M. Vin, Issues in multimedia server design, *ACM Computing Surv*., **27**, 636–639 (1995).
13. H. J. Chen and T. D. C. Little, Storage allocation policies for time dependent multimedia data, *IEEE Trans. Knowledge Data Engng*, **8**, 855–864 (1996).
14. B. Özden, R. Rastogi and A. Silberschatz, On the design of a low-cost video-on-demand storage system, *Multimedia Systems*, **4**, 40–54 (1996).
15. V. O. K. Li, W. Liao, X. Qiu and E. W. M. Wong, Performance model of interactive video-on-demand systems, *IEEE J. Selected Areas Commun*., **14**, 1099–1109 (1996).