Maximizing the Number of Users in an Interactive Video-on-Demand System

Spiridon Bakiras, *Member, IEEE* and Victor O. K. Li, *Fellow, IEEE*

***Abstract*—Video prefetching is a technique that has been proposed for the transmission of variable-bit-rate (VBR) videos over packet-switched networks. The objective of these protocols is to prefetch future frames at the customers' set-top box (STB) during light load periods. Experimental results have shown that video prefetching is very effective and it achieves much higher network utilization (and potentially larger number of simultaneous connections) than the traditional video smoothing schemes. The previously proposed prefetching algorithms, however, can only be efficiently implemented when there is one centralized server. In a distributed environment there is a large degradation in their performance. In this paper we introduce a new scheme that utilizes smoothing along with prefetching, to overcome the problem of distributed prefetching. We will show that our scheme performs almost as well as the centralized prefetching protocol even though it is implemented in a distributed environment. In addition, we will introduce a call admission control algorithm for a fully interactive Video-on-Demand (VoD) system that utilizes this concept of distributed video prefetching. Using the theory of effective bandwidths, we will develop an admission control algorithm for new requests, based on the user's viewing behavior and the required Quality of Service (QoS).**

***Index Terms*—Admission control, distributed video prefetching, effective bandwidth, user interactivity, video-on-demand.**

I. Introduction

THE FAST development of the Internet, and the introduction of new architectures, which can provide a service beyond the traditional best-effort service [1], have made possible the transmission of real-time traffic (e.g., audio and video) that have stringent Quality of Service (QoS) requirements. While internet telephony and video-conferencing are already deployed in the current Internet with limited success, the deployment of applications that require the transmission of high quality video is still lacking. An application like Video-on-Demand (VoD) will allow a customer to select any movie from a video server, view it on his/her screen, and have the ability to perform any type of VCR-like operation [2].

This kind of application is quite attractive, but it can result in very poor network utilization if efficient transmission schemes are not employed. Network utilization is defined as the summation of the individual mean rates of all videos currently transmitted, divided by the service rate (or the link capacity). Video is typically compressed in the Motion Picture Experts Group (MPEG) format. To achieve the best compression rate, the output of an MPEG encoder is very bursty and the corresponding peak to mean ratio is very high. This property of variable-bit-rate (VBR) video makes the provision of deterministic QoS guarantees (in this paper we select the packet loss rate at the local switch as the QoS metric) prohibitively expensive. This is because we will have to allocate enough bandwidth to accommodate the peak rate, in order to assure that there will be no loss at the switch. The alternative is to provide statistical QoS guarantees, that is, we guarantee that the loss rate will not exceed a predefined small value (e.g., 10^{-6}). The challenge in providing statistical QoS is to design admission control algorithms that will accurately estimate the required bandwidth.

Most of the proposed schemes in the literature use a buffer at the customer's set-top box (STB) to smooth the video traffic and, therefore, reduce significantly the peak rate and the rate variability [3]–[5]. Video smoothing can provide both deterministic and statistical QoS guarantees, with the latter being more desirable as it offers higher network utilization. In [5], for example, a simulation study showed that the optimal smoothing algorithm can support, under deterministic service and for a buffer size of 256 KBytes, 185 smoothed *Star Wars* streams (with an average rate of 0.37 Mbps) on a 155 Mbps link, a utilization of 44%, while for statistical service, 304 streams can be supported (without any loss) for a utilization of 73%. The authors also provided an admission control algorithm, assuming a bufferless switch and based on large deviation techniques, which was shown to be quite accurate.

Video prefetching is an alternative technique that has been proposed for the transmission of VBR video, and it can only provide statistical QoS guarantees. In [6] a protocol called Join-the-Shortest-Queue (JSQ) prefetching is presented which has many advantages compared to typical video smoothing schemes. It achieves very high network utilization, facilitates user interactions, and has minimal start-up latency. Their experimental results showed that JSQ prefetching has a loss probability several orders of magnitude smaller than optimal smoothing [5] for the same buffer size and network utilization. The main idea is to put a buffer in the customer's STB which can be used to prefetch future frames when the transmission link is under-utilized. The frames are prefetched in a way such that all the ongoing connections have similar number of prefetched frames. This protocol, however, has a major drawback: it can only be implemented when there is one centralized server which serves different users over a common link. In [7] a decentralized version is

Manuscript received April 17, 2002; revised July 22, 2002. This work was supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project AoE/E-01/99).

The authors are with the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong (e-mail: (sbakiras@eee.hku.hk; vli@eee.hku.hk).

Digital Object Identifier 10.1109/TBC.2002.806194

introduced which allows prefetching when there are many distributed servers in the system. Each server keeps a send window which is the number of frames that it is allowed to send in one frame period. The value of the send window is increased when the sent frames are acknowledged from the user, and it is set to one when frames are dropped. When a frame is dropped, it will be retransmitted from the server if the corresponding client has one or more frames buffered at the STB. This scheme works well compared to other VBR schemes, but its loss probability is more than two orders of magnitude larger than JSQ prefetching for the same buffer size and network utilization. In addition, since there is no coordination between the different servers, this protocol will result in unnecessary retransmission of frames which will increase the traffic load of the core network.

In this paper, we introduce a VBR transmission scheme that utilizes smoothing along with prefetching to reduce the bandwidth requirements of MPEG traffic. The idea is to first smooth the MPEG traces over a few group of pictures (GOPs) and then use a central controller to coordinate the transmission of frames from all the servers [8]. This coordination is possible since the traffic from each connection (video) is constant for a period of time equal to the number of smoothed GOPs (one GOP is normally 12 or 15 frames). We compare our proposed scheme with JSQ prefetching and the results indicate that our scheme performs almost as well as JSQ prefetching even though it is implemented in a system with distributed servers.

The performance of any video prefetching algorithm, however, is very sensitive to user interactions, such as temporal jumps, as all the prefetched frames of the user issuing an interaction request will have to be discarded. The initial work on video prefetching [6], [7] was only based on simulation results, and no analytical model was proposed for the admission of new requests. In this paper we will introduce a call admission control algorithm which will decide on the admission of new requests, based on the user's viewing behavior and the required QoS. In particular, we will use the theory of effective bandwidths [9] to calculate the effective bandwidth for a number of connections, which will depend on the individual traffic parameters, the STB buffer size, the user activity model, and the required QoS. The QoS criteria was considered to be the desired packet loss rate. We will show that the effective bandwidth approach is very accurate, and it adapts very well to different system parameters, such as the level of interactivity. We used 10 MPEG-1 traces [10] to feed our analytical model, and our results indicate that video prefetching is very effective even in an environment with very frequent interaction requests.

The rest of this paper is organized as follows. In Section II we describe the VoD network architecture that is considered throughout this paper, and briefly discuss the basic principles of video prefetching. In Section III we introduce in detail our proposed distributed video prefetching protocol, and also present some experimental performance comparison with the centralized JSQ prefetching protocol. Section IV describes the overall system model, including the video traffic model that was used to model each video source, and our assumptions regarding the user activity model. In Section V we develop the analytical model for the call admission control algorithm, and in Section VI we present our simulation results. Section VII concludes our work.

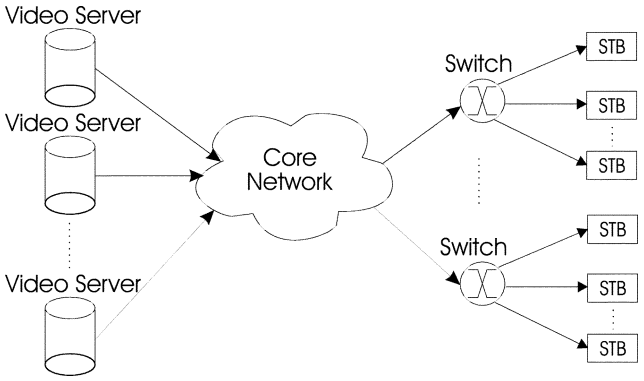


Fig. 1. The network architecture.

II. The VoD Network Architecture

Consider the network architecture shown in Fig. 1. Distributed video servers are connected to the network. These servers may belong to the same or different VoD service providers. The clients are connected to the network through a switch which may, for example, be an Internet Service Provider (ISP) router. Each client has an STB for video decoding which also includes the buffer used for prefetching. When a client makes a request for a particular video, an admission control module (which is located at the local switch) will decide whether that request can be accepted without violating the targeted QoS of the existing connections. If the request is accepted, a connection will be established between the server and the client through the core network. We assume that a reservation protocol is implemented inside the core network to reserve resources for that request. In this work we do not consider the problem of admission control inside the core network, but we only concentrate on the local switch where the users have access to the network. In other words, we consider a VoD-like application where the only type of traffic at the output of the local switch will be stored video. In this case, the admission decision (for the local switch) will be made by the VoD service provider. Inside the core network the different connections between the video servers and the clients will follow different routes and they will be multiplexed with other types of traffic. Therefore, the admission control algorithm will be a more general one which will depend on the core network architecture (i.e., the VoD service provider will not be involved). Multiple clients will have access to the video servers through several different switches. Video prefetching tries to maximize the number of clients served by one such switch, assuming that all the clients connected to that switch will be allowed to share a maximum amount of bandwidth C (e.g., 45 Mbps). It is clear that by doing so, the overall network utilization will be maximized. We assume that the switches in Fig. 1 are bufferless, that is, all packets that exceed the capacity C are dropped.

Before going any further, we should describe briefly the structure and the types of frames of an MPEG sequence. There are three types of frames generated by an MPEG encoder: intraframes (I), predictive frames (P), and bi-directional frames (B) [11]. The I-frames are coded independently of other frames, and for that reason they are used for random access.

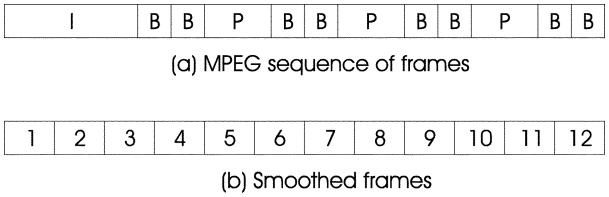


Fig. 2. Smoothing of one GOP at the video server.

The P-frames are coded with respect to a previous I/P-frame, so in general they are smaller than I-frames. Finally, the B-frames are coded with respect to a previous and a future I/P-frame. B-frames are usually much smaller than I- or P-frames. A number of frames, typically 12 or 15, are grouped together to form a group of pictures (GOP). Each GOP has a regular pattern, for example, $IBBPBBPBBPBB$. The GOP is defined by two parameters: the number of frames, q, and the number of B-frames between two consecutive I/P-frames, $l-1$. In the above example, $q = 12$ and $l = 3$.

In our scheme we smooth the traffic from k GOPs before sending it to the core network. So in every $k \cdot q$ consecutive frame periods, the same amount of data, which is equal to the average of the $k \cdot q$ frames, is sent from the server to the core network. This is illustrated in Fig. 2, for the case where $k = 1$, $q = 12$ and $l = 3$. From this point on when we say *frame*, we will refer to a smoothed frame and not to a whole frame of the MPEG sequence. Since we smooth $k \cdot q$ MPEG frames at each time, we should always have some frames buffered at the STB in order to avoid playback starvation. The I-frame of each GOP, for example, will be sent in parts during a few frame periods, and at the point of its display a few frames will be removed from the STB buffer. For this reason, we also need to pre-load the buffer with a few frames prior to the beginning of the playback (start-up latency). In order to guarantee no playback starvation, we should always keep $k \cdot q$ frames in the STB buffer. Note, that the smoothing function is only performed in order to keep the level of traffic constant for a short period of time, and it is not related to the structure of the MPEG-1 sequence. A similar smoothing technique (e.g., smooth every 12 frames) could be implemented for other video formats as well, where the frame pattern is not so regular (e.g., MPEG-4).

The objective of a prefetching protocol is to send additional frames to the different clients when the transmission link is under-utilized. The additional frames are buffered at the STBs, and the prefetching protocol ensures that all customers have similar number of prefetched frames. To demonstrate the underlying principle of video prefetching, let us consider the output bit-rate (Fig. 3) from a number of video connections when each connection is sending one frame per frame period (e.g., without prefetching). There will be some periods where the aggregate bit-rate will be less than the link capacity C, and some periods where it will exceed the link capacity. If we want to keep the loss rate small, we need to place a buffer at the local switch to hold the packets that can not be transmitted on time. This method is presented in detail in [12] where the authors describe and simulate several proposed admission control algorithms. However, the maximum utilization that could be obtained from any of those algorithms was around 85% (which was very close to the maximum obtainable utilization). Video prefetching, on the other hand, can offer a utilization of almost 100% without any need of buffering at the switches. This is achieved by sending additional frames to the clients when the output bit-rate is less than the link capacity (i.e., prefetching). The additional frames will be used to avoid playback starvation when the bit-rate exceeds the link capacity and some frames can not be transmitted on time. In other words, the frames that would have to be buffered at the switch under a nonprefetching scheme, are sent in advance to the clients so that the aggregate bit-rate at the switch will never exceed the allocated bandwidth. We can, therefore, assume that there is a large virtual buffer of size B placed at the local switch, which is physically distributed among the several STBs (Fig. 4). The average size of the virtual buffer will then be

$$B = \sum_{i=1}^{N} (B_i - k \cdot q \cdot m_i) \tag{1}$$

where N is the number of active connections, B_i is the STB buffer size for connection i, and m_i is the mean frame size for connection i. The buffer occupancy $Q(t)$ of the virtual buffer at time t will be

$$Q(t) = B - \sum_{i=1}^{N} b_i(t) \tag{2}$$

where $b_i(t)$ is the buffer level for connection i at time t. The prefetching algorithm tries to keep the STB buffers as full as possible or, in other words, keep the buffer occupancy $Q(t)$ as low as possible. This is the main difference between video prefetching and typical video smoothing schemes. In a video smoothing scheme the STB buffer is only used to smooth the video traffic (i.e., reduce the peak rate and the rate variability), and during some periods of time it can be almost empty.

III. Distributed Video Prefetching

We will now present in detail our proposed prefetching protocol. It is based on the idea of a central controller that coordinates the transmission of frames from the different servers. The local switch (Fig. 1) will be the location of the central controller, since all the video servers are connected to it through the core network. This coordination is possible because of the smoothing that results in a constant level of traffic from each connection for a few frame periods. We will now introduce the following variables associated with each video connection.

- f: frame rate (e.g., 24 frames/sec).
- k: number of GOPs to be smoothed.
- q: GOP size.
- $k \cdot q$: minimum number of frames that should be buffered at the STB at all times, in order to avoid playback starvation (as explained previously).
- g_i: index showing which frame of the smoothed GOP is currently being transmitted to customer i. It takes the values $1, 2, \ldots, k \cdot q$.
- b_i: current buffer level for customer i.
- B_i: maximum buffer size for customer i. For simplicity we assume that all customers have the same buffer size.

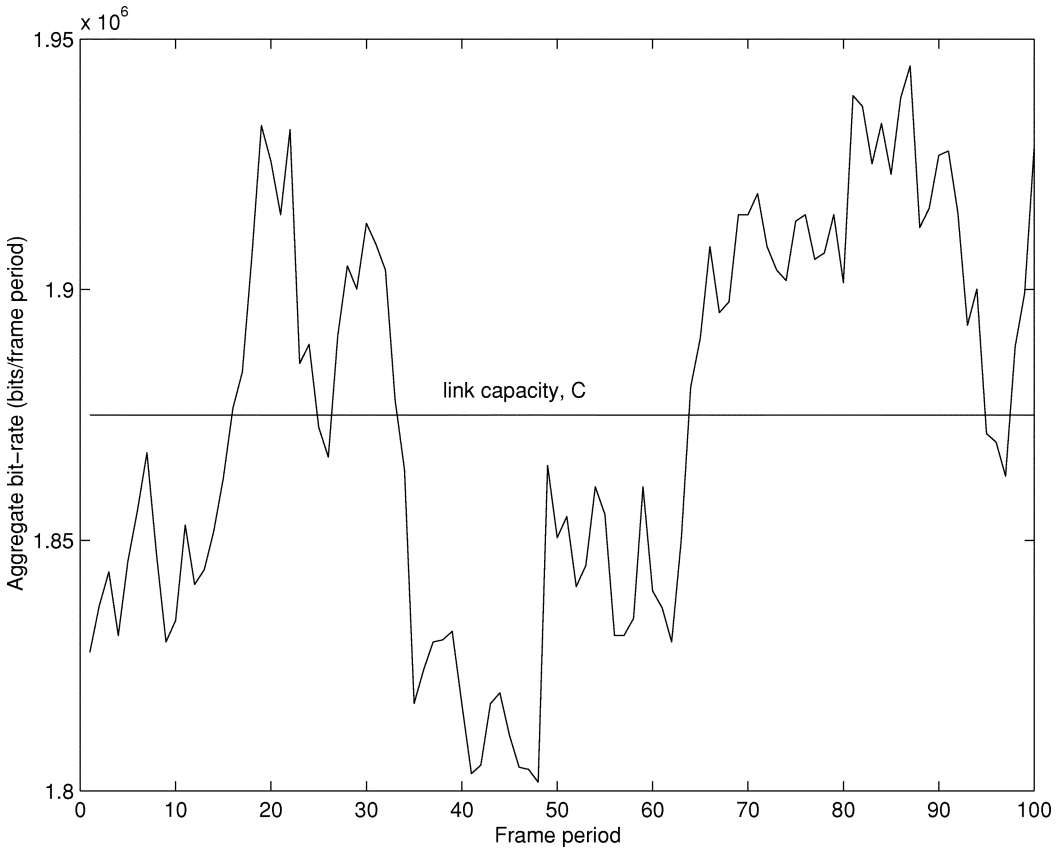


Fig. 3. Output bit-rate without prefetching.

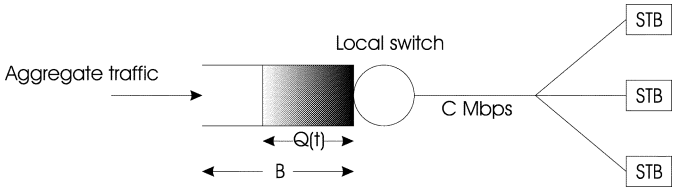


Fig. 4. The virtual buffer at the local switch.

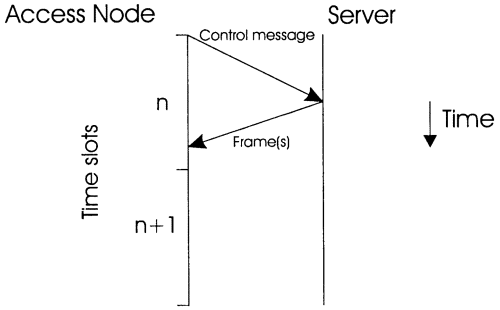


Fig. 5. Time slot synchronization between a server and the central controller.

- w_i: number of frames buffered for customer i.
- p_i: number of prefetched frames for customer i. It is equal to $\max\{w_i - k \cdot q, 0\}$.
- r_i: maximum number of frames that can be sent in the following frame period to customer i.
- l_i: number of frames to be sent in the following frame period to customer i.
- s_i: size of the frame transmitted to customer i in the current frame period.

Let us call a *time slot* the time period corresponding to one frame which is equal to $1/f$ seconds. During this time slot, the central controller will transmit the frames from the different servers to the clients, until the link capacity C is reached. If some frames can not be transmitted in the current time slot, they will be discarded. A discarded frame will be retransmitted from the server if the corresponding connection has $p_i \geq 1$, which means that this connection has some frames buffered at the STB.

Since the frames will be transmitted from the controller to the clients according to the controller's own discrete time slots, there must be some kind of synchronization between all servers and the controller. We will assume that the round trip propagation and processing delay between any server and the local switch is less than a time slot (i.e., $1/f$ seconds). The controller will coordinate the transmission of frames from all the video servers by sending control messages to them. The control messages will indicate which frames should be sent for each connection. All the control messages will be sent in the beginning of the current time slot, say n (Fig. 5) and, therefore, the frames from all connections will arrive prior to the beginning of time slot $n+1$. They will, then, be transmitted to the clients within the duration of time slot $n+1$. This will allow the controller to know the exact frame sizes to be transmitted in slot $n+2$ (because of the GOP smoothing), and based on this information it will send the appropriate control messages at the beginning of slot $n+1$. Note, however, that this delay bound does not have to be tight. Even if some frames arrive after the beginning of time

slot $n+1$, and the controller will not have the exact frame sizes of the frames to be transmitted in slot $n+1$, the prefetching algorithm can use the corresponding information from the previous transmission (slot n) as an estimate. Due to the high correlation between the frame sizes of consecutive GOPs, the impact of this approximation on the performance of our scheme will be insignificant.

A. The Prefetching Algorithm

Let us consider the transmission schedule of five videos shown in Fig. 6 where we assume that we smooth one GOP ($k = 1$) with $q = 12$, prior to the transmission. This corresponds to the frames that would be transmitted from the local switch to the customers if no coordination took place. The numbers in each box are the g_is for the different connections. Therefore, in time slot n, the first frame of the smoothed GOP would be transmitted to customer 1, the seventh frame to customer 2, and so on. By the beginning of slot n the controller will already know the frame sizes for all five connections, and it will use this information to coordinate the transmission of frames from the video servers for the following time slot.

Since we want all connections to have similar number of prefetched frames, the controller will try to prefetch frames from all connections in order of ascending values of p_i (as in the JSQ prefetching protocol). Let us assume for simplicity that all connections in Fig. 6 have $p_i = 0$. The controller will first try to prefetch one frame from connection 1. Since the transmission will take place in time slot $n+1$, the maximum number of frames that can be sent for connection 1 is 11. In the general case, the maximum number of frames that can be sent for one connection is $r_i = \max\{k \cdot q - g_i, 1\}$, since these are the only frames for which we know their exact size. Note that in the case where $g_i = k \cdot q$ the server will only be allowed to send one frame, since the size of the next $k \cdot q$ frames is not known until the first one of them arrives at the controller. Let us call W the estimate of the amount of traffic that will be sent to the local switch in time slot $n + 1$ (initially $W = 0$). The number of frames to be sent for each connection is set initially to $l_i = 0$. There are two factors that can limit the number of frames to be sent: the maximum buffer size B_i and the link capacity C (in packets/frame period). The controller will check whether the following two conditions hold:

$$b_i + (l_i + 1) \cdot s_i \leq B_i \tag{3}$$

$$W + s_i \leq C. \tag{4}$$

Equation (3) tries to avoid buffer overflow while (4) checks whether the additional frame will keep the value of the estimate of the total traffic in time slot $n + 1$ below the link capacity. In (3), we do not consider the MPEG frame that will be removed from the buffer during time slot $n + 1$, since it is not possible for the controller to have this information at the time when the prefetching algorithm is executed. We assume, however, that the controller knows the buffer level b_i for each connection (this can be done with control messages from the STBs). If both conditions hold for connection 1, the controller will update the values of l_1, p_1, r_1, and W, and it will continue with connection 2. The algorithm will stop when there is no connection with $r_i > 0$ that satisfies both (3) and (4). When the algorithm terminates, the controller will check whether there is any connection with $p_i = 0$ for which the algorithm returned the value $l_i = 0$. If such a connection is found, l_i will be set to one, since the next frame has to be transmitted in order to meet its deadline. When the values of all l_is have been updated, the controller will send the control messages to the corresponding servers at the beginning of time slot n. The control messages will be sent only for those connections with $l_i > 0$. The complete prefetching algorithm is presented in Fig. 7.

Time slot: n n+1 n+2 n+3 n+4 .....

Video 1	1	2	3	4	5	6	7	8	9	10	11	12
Video 2	7	8	9	10	11	12	1	2	3	4	5	6
Video 3	10	11	12	1	2	3	4	5	6	7	8	9
Video 4	12	1	2	3	4	5	6	7	8	9	10	11
Video 5	4	5	6	7	8	9	10	11	12	1	2	3

Fig. 6. Transmission schedule without prefetching.

```
procedure prefetch()
begin
  W := 0;
  for all connections i do
  begin
    r_i := max{k · q − g_i, 1};
    l_i := 0;
    p'_i := p_i;
    e_i :=false;
  end;
  while exists i with e_i=false do
  begin
    find connection i with minimum p'_i;
    if {(3)=true and (4)=true and r_i > 0} do
    begin
      l_i := l_i + 1;
      p'_i := p'_i + 1;
      r_i := r_i − 1;
      W := W + s_i;
    end;
    else
      e_i :=true;
  end;
end.
```

Fig. 7. The prefetching algorithm.

B. Transmission of Frames

The frames are transmitted from the local switch to the clients based on a nonpreemptive priority scheme. Connections with smaller values of p_i have priority over those with bigger values of p_i. The nonpreemptive scheme will allow all the frames from one connection to be transmitted, once it has started the transmission. The value of p_i will be updated by the controller as soon as it receives the frames from all connections, through the following equation

$$p_i = \max\{p_i + f_i - 1, 0\} \tag{5}$$

where f_i is the number of frames that were successfully transmitted to customer i. Since all the frames will arrive by the beginning of the time slot, the controller can easily calculate, based

on the priorities and the frame sizes, which frames will be transmitted in the current time slot.

As we mentioned earlier, the value of W used in the prefetching algorithm is an estimate of the total traffic at the time slot where prefetching will occur. However, many connections will probably have frames from different GOPs being transmitted (i.e., the first frame of the next GOP), so the actual amount of traffic in this time slot might be higher or lower than W. If it is higher, then some frames will have to be dropped since W will probably be very close to the link capacity. The transmission order of our protocol will try to drop frames from those connections that have $p_i \geq 1$. If a frame from such connection is dropped, the controller will indicate to the corresponding server, through the next control message, to retransmit the frame.

To conclude the presentation of our protocol, we should describe briefly the operation of the server. Each server remains idle until it receives a control message. When it receives the control message, it will transmit the indicated frame(s). In order to avoid buffer overflow, the server will check the following condition before sending the first frame of a smoothed GOP (i.e., the frame with $g_i = 1$)

$$b_i + s_i \leq B_i. \tag{6}$$

This is necessary, since the value of s_i that was used in (3) was from the previous smoothed GOP, and its current value could be much higher. If condition (6) holds it will send the frame, otherwise it will remain idle. The buffer level for each connection can be included in the control message, as we have assumed that the controller has this information.

C. Comparison With JSQ Prefetching

In order to evaluate the effectiveness of our scheme, we used 10 real MPEG-1 traces that were downloaded from the public domain [10]. They covered a wide variety of contents, including movies, news, talk shows, sports, music, and cartoons. All the traces were captured at $f = 24$ fps and the GOP parameters were $q = 12$ and $l = 3$. Even though these traces have some problems (e.g., some frames are dropped), they are very bursty and they exhibit self-similarity, which makes them suitable for our simulations. The total number of frames for each trace was 40 000 which is approximately 30 minutes in duration. We assumed that the capacity C is 45 Mbps, and that the frames are transmitted from the video servers to the clients in fixed size packets of 1 Kbit. In Table I we have summarized some characteristics of the different MPEG traces.

In each experiment we used a mixture of traces that resulted in a 99% network utilization, that is, the summation of the individual average bit-rates was approximately 99% of the link capacity (45 Mbps). This mixture consisted of 8 *Asterix* traces, 9 *Tennis* traces, 11 *Mr. Bean* traces, 8 *James Bond* traces, 14 *Jurassic Park* traces, 9 *Mtv* traces, 12 *News* traces, 5 *Race* traces, 8 *Soccer* traces, and 13 *Talk Show* traces, for a total of 97 connections. The experiments were performed as follows. For each connection we chose a random starting point in the movie (the beginning of a GOP), and we started by transmitting one frame from each connection, with all connections having $b_i = 0$. From the next time slot the prefetching algorithm was used to coordinate the transmissions until the end of the experiment. When a connection displayed the last MPEG frame of the movie, the same movie started again from a new random starting point (with $b_i = 0$). But from this point on, we used appropriate wrap-around such that each connection displayed exactly 40 000 frames. We simulated 10^8 frame periods for several buffer sizes, ranging from 256 to 896 KBytes in increments of 128 KBytes. We run the experiments using the same random seed for the two protocols, so as to ensure a fair comparison between them. Finally, we counted the time slots that experienced losses after an initial period of 10 000 frames (to allow the buffers to fill up).

TABLE I
CHARACTERISTICS OF THE MPEG-1 COMPRESSED VIDEO SEQUENCES

Sequence	Mean (bits)	Peak/Mean
Asterix	22,348	6.6
ATP Tennis	21,890	8.7
Mr Bean	17,647	13
James Bond: Goldfinger	24,308	10.1
Jurassic Park	13,078	9.1
Mtv	19,780	12.7
News	15,358	12.4
Race	30,749	6.6
Soccer	25,110	7.6
Talk show	14,537	7.3

In Fig. 8 we have plotted the loss probability as a function of the buffer size, for the JSQ prefetching protocol and our proposed scheme. We simulated two versions of our protocol for the cases of $k = 1$ and $k = 3$. It is clear that the performance of our protocol is very similar to the performance of the JSQ prefetching protocol. For $k = 1$, the loss probability of our scheme is only one order of magnitude bigger than the loss probability of the JSQ prefetching protocol. In fact, our protocol has the same loss probability as the JSQ prefetching protocol, with a buffer size increment of only 128 KBytes. JSQ prefetching had zero loss for a buffer size of 796 KBytes while, for $k = 1$, our scheme had zero loss for a buffer size of 896 KBytes (in Fig. 8, however, we used the value of 10^{-10} to represent zero). The decentralized prefetching protocol described in [7] had a loss probability of more than two orders of magnitude bigger than the JSQ prefetching protocol and, in addition, its loss probability decreased very slowly with increasing buffer size. The performance of our scheme for $k = 3$, is slightly worse than the one for $k = 1$. One would expect that smoothing three GOPs would result in a better performance, since the prefetching algorithm would be more efficient as the traffic is constant for longer periods of time. However, the disadvantage of using $k = 3$ is that we need a larger buffer size to hold the $k \cdot q$ frames at all times, so the performance does not improve compared to the case where $k = 1$. For the remainder of this work we will, therefore, assume that smoothing is always performed over one GOP.

In Fig. 9 we have plotted the number of frames that are retransmitted as a result of excess traffic at the local switch. These

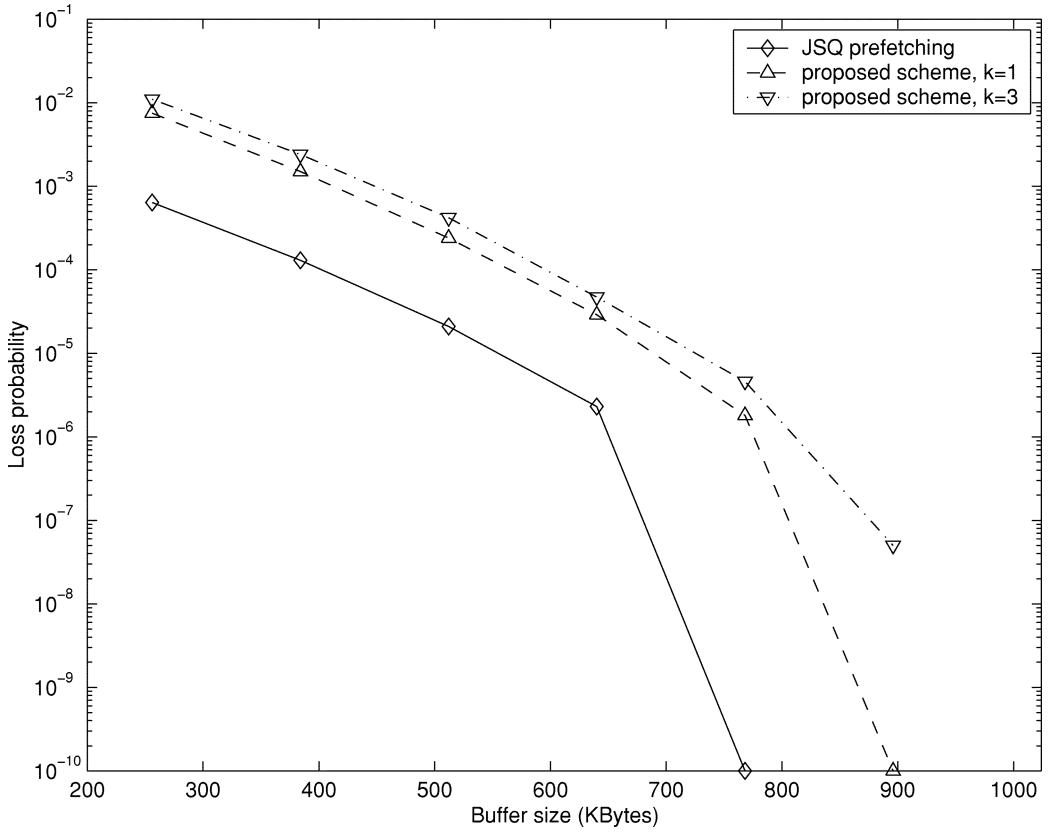


Fig. 8. Loss probability for different buffer sizes.

retransmissions are caused by the incorrect estimation of W in (4). It is obvious that these retransmissions are very rare, and only a few frames (from any of the 97 connections) are dropped at any instant. In particular, throughout our experiments, an average of 0.5 frames per frame period were dropped. The decentralized protocol proposed in [7] offers no coordination between the different servers, and it will lead to a lot more retransmissions, thus increasing the traffic load of the core network. In addition, the send window used in [7] has a minimum value of one which means that all servers send at least one frame at each frame period. But at such high network utilization (99%) this will certainly cause many retransmissions. In our scheme, however, no frames will be transmitted for those connections that the prefetching algorithm returned the value $l_i = 0$.

IV. System Model

A. The Video Traffic Model

There have been many studies in the literature that indicate the self-similar nature of network traffic (e.g., [13]). A self-similar process is characterized by a slowly decaying autocorrelation function, which implies that traffic bursts are present in multiple time-scales (ranging from milliseconds up to minutes). Furthermore, other researchers have shown (e.g., [14], [15]) that VBR video traffic is self-similar. Following these guidelines, we will use a three parameter model to properly characterize the traffic that is generated by a video sequence. These parameters have been introduced by Norros in [9], and they will be used in the next section to calculate the effective bandwidth for the aggregate video traffic. Specifically, each video sequence i will be modeled by the following three parameters:

- The mean rate m_i in bps.
- The variance coefficient a_i in bits-sec.
- The Hurst parameter H_i, where $0.5 < H_i < 1$.

These parameters may be calculated off-line using certain estimators that have been proposed in the literature (e.g., [16]).

When N video sources are multiplexed over a common link, the corresponding parameters of the aggregate traffic are given by

$$m = \sum_{i=1}^{N} m_i$$

$$a = \frac{\sum_{i=1}^{N} m_i a_i}{m}$$

$$H = \max_i \{H_i\}. \tag{7}$$

In an interactive system, though, the user will be able to perform any kind of VCR-like functions. The only function that will affect the traffic generated from the video server for the particular connection, is the fast playback (i.e., fast forward, fast reverse). There are many ways in which we can display a video sequence at a faster rate. For example, we can send the same sequence, but increase the display rate (e.g., 90 fps, instead of 30 fps). The disadvantage of this method is that the display device might not be able to support such a high display rate. The alternative is to skip a number of frames during the display, and

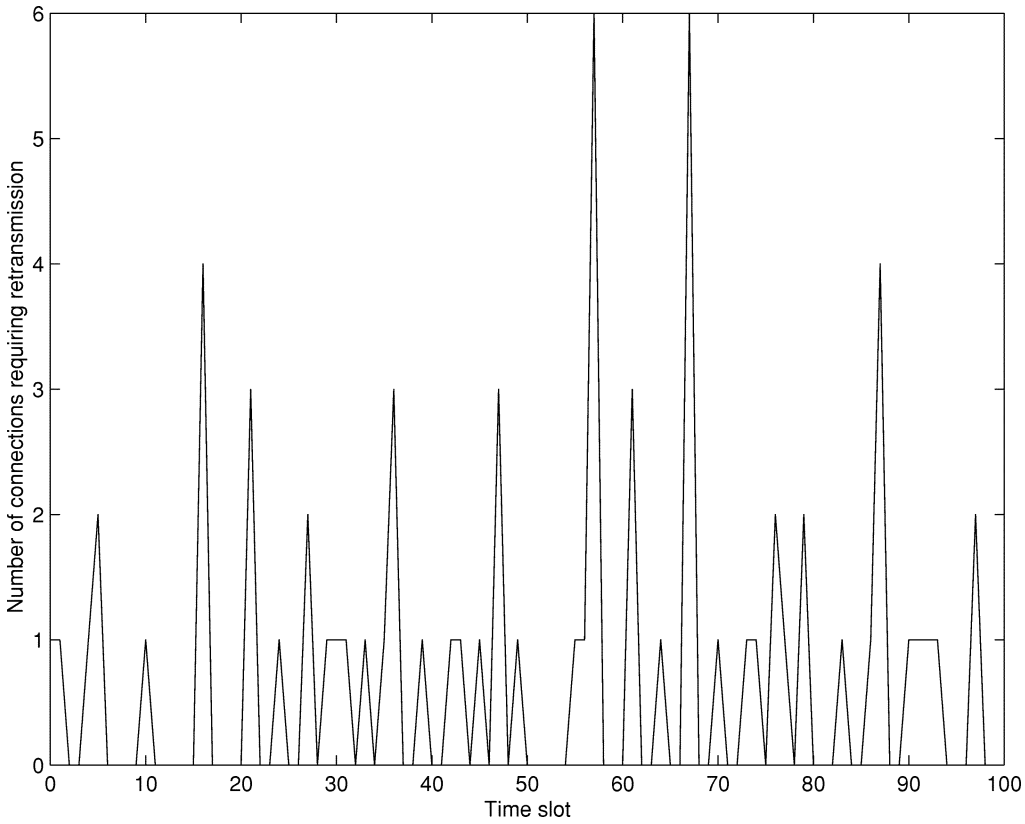


Fig. 9. Number of frames requiring retransmission ($k = 1$).

keep the same display rate. In this work, we selected this second method to implement the fast playback operation.

In our protocol, during a fast playback operation we will skip all the B-frames of the MPEG sequence, and send only the I- and P-frames. Since the P-frames require only the previous I/P-frame for decoding, all the transmitted frames will be decodable at the STB. The video traces that we used in our simulations were captured at 24 fps, and each GOP had the pattern $IBBPBBPBBPBB$. In this case, when a user initiates a fast forward request, the video server will send only 4 frames per GOP and they will be displayed at 24 fps. To the user, it will seem like the display rate is 3 times faster. The I- and P-frames of each GOP will first be smoothed at the video server so as to reduce the variability of the resulting traffic. We can then use the same traffic model for the case of fast playback, by considering only the I- and P-frames of the video trace. The resulting three parameters $(m_i^{IP}, a_i^{IP}, H_i^{IP})$ will model the traffic generated by connection i when it is in the fast playback mode.

B. User Interactivity

In an interactive VoD system each user will be able to perform any type of VCR-like functions, at any time. As we will see in the following section, serving an interaction request requires additional system resources (i.e., more bandwidth) and, therefore, we should take this fact into account when designing the admission control algorithm. There are two ways to perform admission control in an interactive system: we can either perform admission control each time a new request or an interaction request arrives (with interaction requests having priority over the new), or reserve some amount of bandwidth for the interaction requests during the admission control of new requests. We believe that the latter is more suitable for a VoD-like application, since an increased blocking probability of new requests is more desirable than an increased blocking probability of interaction requests.

To properly account for the effect of user interactions, we need a user activity model. Without loss of generality, in this paper, we assume that each user follows the two-state activity model proposed in [17]. In this model, the user starts in normal playback state, and stays there for a period of time which is exponentially distributed with mean $1/\lambda$. He then moves to the interaction state where he will issue an interaction request. He will stay in the interaction state for a period of time which is again exponentially distributed with mean $1/\mu$, and move back again to the normal playback state. This will be repeated until the end of the video sequence. The parameters λ and μ are the interaction arrival and service rates, respectively. In order to perform admission control in such a system, we need to consider each type of user interaction separately. This will be the subject of the following section. Note that we will not consider any *pause/stop* operations in our analysis. If the admission control algorithm admits more connections based on the assumption that some of them will be paused or stopped at any time, the required QoS will be violated if this assumption turns out to be optimistic.

V. Admission Control

For the admission control algorithm we will use the effective bandwidth formula provided by Norros in [9]. This formula is derived based on the assumption that the buffer size in very

large, which is normally not true for commercial switches. In video prefetching, however, the virtual buffer size is sufficiently large (e.g., for 100 clients with 1 MB buffer each, the buffer size B is 100 MB), and this approximation is quite accurate. In the next section we will show, through simulation experiments, that the effective bandwidth approach is indeed very accurate when utilized in a prefetching scheme.

Let us call C_e the effective bandwidth of the aggregate traffic, and C the total available bandwidth. Assume that there are currently $N-1$ connections in progress, and there is a request for a new connection. The admission control algorithm will calculate the new C_e from the following equation

$$C_e = m + \left(\kappa(H)\sqrt{-2\ln(\epsilon)}\right)^{1/H} a^{1/2H} B^{-((1-H)/H)} m^{1/2H} \tag{8}$$

where $\kappa(H) = H^H(1-H)^{(1-H)}$, ϵ is the desired loss rate, and the parameters $(m,\ a,\ H)$ are calculated as in (7). If $C_e \leq C$ the new connection will be admitted; otherwise, it will be rejected. Notice that the buffer size B will have to be re-calculated, since it depends on the total number of connections N [see (1)]. The above admission control algorithm is applicable only in a system that does not support user interactions. In the following two subsections we will investigate how the different user interactions affect the number of admissible connections.

A. Fast Forward/Reverse

As we mentioned in Section IV-A, a *fast rate* request issued by a client will increase significantly the amount of traffic sent for that particular client, since only the I- and P-frames will be sent. Therefore, we should reserve some amount of bandwidth during admission control, so that subsequent *fast rate* requests will not violate the QoS requirements of any client. The extreme case would be to reserve enough bandwidth to accommodate the scenario where all the clients are on *fast rate* mode simultaneously. Obviously this is a very conservative approach, and the resulting network utilization would be very poor. The alternative is to reserve less bandwidth, and then reject some interaction requests according to some rule. More specifically, we will reserve an amount of bandwidth which will be able to accommodate n concurrent *fast rate* operations, while maintaining a low blocking probability for interaction requests. Let us call λ_f the arrival rate of *fast rate* requests, and μ_f the corresponding service rate. Since both parameters are exponentially distributed according to our user activity model, we can model this system as an $M/M/n/n/N$ queueing system, that is, an n-server loss system with finite customer population N. We are interested in finding a number n, such that the stationary probability p_n is less than a small number, where n is the number of customers in the system (i.e., the number of users served in *fast rate* mode simultaneously). The desired value of p_n will be set by the VoD service provider, according to their policy. In this work we will assume that $p_n \leq 0.01$. The formula for p_n is easy to derive and it is given by [18]

$$p_n = \frac{\binom{N}{n}\left(\frac{\lambda_f}{\mu_f}\right)^n}{\sum_{i=0}^{n}\binom{N}{i}\left(\frac{\lambda_f}{\mu_f}\right)^i}. \tag{9}$$

Then, n will be the smallest integer that satisfies the inequality $p_n \leq 0.01$.

The three parameters of the aggregate traffic may then be approximated as follows:

$$\begin{aligned} m_f &= \frac{(N-n)}{N}\sum_{i=1}^{N} m_i + \frac{n}{N}\sum_{i=1}^{N} m_i^{IP} \\ a_f &= \frac{(N-n)}{N}\frac{\sum_{i=1}^{N} m_i a_i}{m} + \frac{n}{N}\frac{\sum_{i=1}^{N} m_i^{IP} a_i^{IP}}{m} \\ H_f &= \max_i\{H_i,\ H_i^{IP}\}. \end{aligned} \tag{10}$$

The admission control algorithm will be based on the assumption that there will always be n users in *fast rate* mode. The system will also keep track of the number of connections that are in *fast rate* mode at all times, and it will block an interaction request if this number is equal to n. The assumption of constantly having n users in *fast rate* mode will clearly overestimate the required bandwidth. It is essential, though, to make this assumption, in order to guarantee the targeted QoS during periods of intense user activity.

When a customer initiates or terminates a *fast rate* request, all the prefetched frames in the STB buffer will have to be discarded, since they are no longer useful. As the average service time of such requests will normally be very small (around 10–20 seconds), it is not efficient to fill up the STB buffer again with future frames. We will, therefore, assume that only one frame per frame period is sent to a connection that operates in *fast rate* mode. As a result, the buffer size B initially given in (1) will now be equal to

$$B = \frac{(N-n)}{N}\sum_{i=1}^{N}(B_i - k\cdot q\cdot m_i) \tag{11}$$

where N is the total number of ongoing connections, including the new request.

The idea of reserving some amount of bandwidth for accommodating *fast rate* requests was first proposed in [19] where the authors considered two different approaches. In the first one, a *fast rate* request is delayed until there are available resources, and the admission control ensures that the probability that this delay exceeds a certain value is small. This approach can be implemented in our scheme as well, if we allow the interaction requests to be queued up instead of blocking them. In the second approach, there is no delay associated with an interaction request, but when there are not enough system resources to serve all the interaction requests, the picture quality of the users in *fast rate* mode is degraded. However, for a VoD system to be competitive with the existing video rental services, it should offer a better service to the user. In our scheme, the picture quality is never degraded, and the system response to user interactions is instantaneous. The blocking probability p_n can also be set to a very small value, practically eliminating blocked requests. In addition, the work in [19] considered peak rate bandwidth allocation for each connection, leading to low network utilization. In our scheme, we employ video prefetching and statistical

multiplexing which can increase significantly the network utilization. This is basically the main contribution of our work. To our knowledge, there is no admission control scheme in the literature that can be directly applied in a real VoD system. In this work we propose a complete solution which considers all the aspects of a real system: transmission protocol (i.e., prefetching), user interactivity, and statistical multiplexing for VBR video.

B. Jump Forward/Backward

In typical video transmission schemes, *jump* operations do not affect the network utilization. In video prefetching, however, frequent *jump* requests will degrade the utilization of the system. Suppose a user initiates a *jump* request during normal playback. Since this operation will take the user to a point in the video sequence which will be quite far (where far means anything more than 10–15 seconds) from the current point, all the prefetched frames will have to be discarded, as they will not be displayed. It is clear that when those requests are frequent, the buffer size B will decrease and, thus, the effective bandwidth for the same connections will increase.

Modeling the effect of *jump* requests on the buffer size B is not easy. We may model, however, the event of buffer loss by assuming that a *jump* request will trigger the arrival of additional traffic at the switch, which is equal to the average buffer size of a connection. Let us call λ_j the arrival rate of *jump* requests. The additional amount of traffic will clearly be Poisson, since interaction requests arrive according to a Poisson process. The corresponding parameters for the Poisson traffic may be calculated as follows:

- $m_j = N\lambda_j \cdot B/N$.
- For Poisson traffic the variance coefficient is equal to the variance over the mean rate. Therefore, $a_j = 1$.
- The Hurst parameter for Poisson traffic is $H_j = 0.5$.

Finally, we may approximate the traffic parameters of the aggregate traffic as follows:

$$m_{tot} = \frac{(N-n)}{N}\sum_{i=1}^{N} m_i + \frac{n}{N}\sum_{i=1}^{N} m_i^{IP} + m_j$$

$$a_{tot} = \frac{(N-n)}{N}\frac{\sum_{i=1}^{N} m_i a_i}{m} + \frac{n}{N}\frac{\sum_{i=1}^{N} m_i^{IP} a_i^{IP}}{m} + \frac{m_j a_j}{m}$$

$$H_{tot} = \max_i\{H_i, H_i^{IP}, H_j\}. \tag{12}$$

Note, that the above estimation of m_j is somewhat conservative, since we assume that the buffer of a client issuing an interaction request is always filled up. In general this will not hold, especially when the buffer size B_i of the client is relatively large. Therefore, m_{tot} will slightly overestimate the required bandwidth.

C. The Complete Admission Control Algorithm

After analyzing the effect of different types of user interactions on the effective bandwidth, we are ready to present the complete admission control algorithm. The inputs of the algorithm will be the traffic parameters of all ongoing connections (including the new request), the arrival and service rates for the different user interactions, the STB buffer size B_i, and the link capacity C. The admission control will be performed as follows.

1) Calculate n from (9). This value will depend on the number of active users N, including the new request.
2) Calculate the buffer size B from (11).
3) Calculate the traffic parameters from (12).
4) Calculate the effective bandwidth C_e from (8).
5) If $C_e \leq C$ admit the new request, else reject it.

TABLE II
ACTUAL AND PREDICTED NETWORK UTILIZATION (%) WITHOUT USER INTERACTIONS

	$B_i = 128$ KB		$B_i = 1$ MB	
C (Mbps)	Actual	Predicted	Actual	Predicted
45	96.7	97.5	98.7	98.7
155	98.0	98.9	99.5	99.5

VI. SIMULATION RESULTS

In order to investigate the accuracy of the admission control algorithm we used the same video traces that where introduced in Section III-C. The total number of frames for each trace is 40 000, and by using four copies of each trace we created 10 sequences, each of approximately 111 minutes. The experiments were performed as follows. We created a random sequence of requests for different movies, and using the admission control algorithm, a certain number of those requests were accepted. For each of the accepted requests, we chose a random starting point in the movie (the beginning of a GOP), and we started by transmitting one frame from each connection, with all buffers being initially empty. From the next time slot the prefetching algorithm described in Section III-A was used to coordinate the transmissions until the end of the experiment. When a connection displayed the last MPEG frame of the movie, the same movie started again from the beginning, with an empty buffer. Each connection started in normal playback, and stayed there for a period of time which was exponentially distributed with mean $1/\lambda$. Then it moved to the interaction state where it issued an interaction request. The time spent in *fast rate* mode was exponentially distributed with mean $1/\mu_f = 30$ seconds. The requested offset during a *jump* request was uniformly distributed between 1 and 1000 seconds. We simulated 10^8 frame periods for different buffer sizes, and interaction arrival rates λ. Finally, we counted the packet loss rate after an initial period of 50 000 frames (to allow the buffers to fill up). The required loss rate was set to 10^{-6}.

First, we simulated a system without user interactions. The results are given in Table II for different values of STB buffer size B_i, and link capacity C. The actual utilization was obtained by admitting additional connections (up to a utilization level of 100%), and then gradually removing some connections until the QoS requirement was met. For a moderate buffer size of 1 MB the admission control algorithm is very accurate, and predicts exactly the number of connections that should be admitted. In addition, the network is able to work at a utilization level of almost 100%. For a small buffer size of 128 KB there is an underestimation of around 1%, which is due to the fact that the

TABLE III
ACTUAL AND PREDICTED NETWORK UTILIZATION (%) WITH USER INTERACTIONS ($C = 45$ Mbps)

$1/\lambda$	$B_i = 128$ KB				$B_i = 1$ MB			
	Jump		Fast rate		Jump		Fast rate	
(min)	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted
10	91.3	97.5	88.8	88.8	97.5	96.3	93.0	90.3
20	92.5	97.5	90.5	91.3	97.5	97.5	95.2	92.5
30	92.5	97.5	90.5	92.5	98.3	97.5	96.4	93.7
40	92.5	97.5	91.3	93.3	98.3	98.3	97.2	94.6

TABLE IV
ACTUAL AND PREDICTED NETWORK UTILIZATION (%) WITH USER INTERACTIONS ($C = 155$ Mbps)

$1/\lambda$	$B_i = 128$ KB				$B_i = 1$ MB			
	Jump		Fast rate		Jump		Fast rate	
(min)	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted
10	97.0	98.6	93.2	93.2	98.9	96.7	95.0	93.8
20	97.0	98.8	95.2	95.2	99.3	98.0	97.5	95.7
30	97.0	98.9	96.1	96.1	99.5	98.6	98.0	96.7
40	97.0	98.9	96.1	96.4	99.5	98.8	98.7	97.0

derivation of the effective bandwidth formula assumes a very large buffer size. Comparing those numbers in Table II with the utilization achieved by typical video smoothing schemes (i.e., around 73% for the optimal smoothing algorithm [5]), the effectiveness of video prefetching is clearly illustrated.

Next, we tested the accuracy of the admission control algorithm under different types of user interactions. Tables III and IV summarize the results for various interaction arrival rates. These results are very consistent, and they indicate that the admission control algorithm is on the right track. For a buffer size of 1 MB there is a slight overestimation of around 1%–3%, which is caused by some of our assumptions in Sections V-A and V-B (as explained there). Furthermore, the admission control algorithm becomes more accurate when more connections are admitted (i.e., when $C = 155$ Mbps), since the size of the buffer B is significantly increased. Similar observations hold for the case where $B_i = 128$ KB. When the number of admissible connections is small (Table III), the admission control algorithm underestimates the required bandwidth by as much as 6% (due to the small size of buffer B). For a link capacity of 155 Mbps, though, the accuracy of the algorithm is increased, and the underestimation is kept below 2%.

VII. CONCLUSIONS

We have presented a new scheme for the effective transmission of MPEG-compressed video traffic over packet-switched networks, for a VoD system with distributed video servers. It is based on the idea of video prefetching that was originally proposed in [6], [7]. Our motivation was the fact that these protocols are efficient only when there is one centralized server in the system that serves all the clients over a common transmission link. In a distributed environment, there is a large degradation in the performance of the prefetching protocols. In our scheme, we used a central controller to coordinate the transmission of future frames between all the video servers. To make this possible, the MPEG traffic is first smoothed over a number of GOPs before entering the network. Therefore, the central controller is able to coordinate future transmissions, as the traffic will be constant over a number of frame periods. We compared our scheme with the centralized JSQ prefetching protocol, and the simulation results showed that our scheme has very similar performance. However, our scheme has the advantage that it is implemented in a system with distributed video servers, which allows multiple VoD service providers to share the same network.

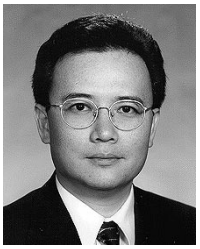
In addition, we have introduced a call admission control algorithm for a VoD system, where each user is allowed to interact at any time during normal playback. The theory of effective bandwidths was used to design the admission control algorithm for a system that supports full user interactivity. We have shown that the proposed algorithm is very accurate and it adapts very well to different system parameters, such as the level of interactivity. The simulation results indicate that video prefetching is very effective and, combined with our proposed admission control algorithm, it can achieve a network utilization of nearly 100%. In addition, it performs very well even in an environment where user interactions are very frequent.

REFERENCES

[1] X. Xiao and L. M. Ni, "Internet QoS: A big picture," *IEEE Network*, pp. 8–18, Mar./Apr. 1999.
[2] V. O. K. Li and W. J. Liao, "Distributed multimedia systems," *Proc. IEEE*, vol. 85, no. 7, pp. 1063–1108, July 1997.

[3] J. M. McManus and K. W. Ross, "Video-on-demand over ATM: Constant-rate transmission and transport," *IEEE J. Select. Areas Commun.*, vol. 14, no. 6, pp. 1087–1098, August 1996.
[4] J. D. Salehi, Z. L. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," in *Proc. ACM SIGMETRICS*, May 1996, pp. 222–231.
[5] Z. L. Zhang, J. F. Kurose, J. D. Salehi, and D. Towsley, "Smoothing, statistical multiplexing and call admission control for stored video," *IEEE J. Select. Areas Commun.*, vol. 15, no. 6, pp. 1148–1166, Aug. 1997.
[6] M. Reisslein and K. W. Ross, "Join-the-shortest-queue prefetching protocol for VBR video on demand," in *Proc. IEEE Int. Conf. Network Protocols (ICNP)*, Oct. 1997, pp. 63–72.
[7] M. Reisslein, K. W. Ross, and V. Verillotte, "A decentralized prefetching protocol for VBR video on demand," in *Proc. 3rd European Conf. Multimedia Applications, Services and Techniques (ECMAST)*, May 1997, pp. 388–401.
[8] S. Bakiras and V. O. K. Li, "Smoothing and prefetching video from distributed servers," in *Proc. IEEE Int. Conf. Network Protocols (ICNP)*, Oct. 1999, pp. 311–318.
[9] I. Norros, "On the use of fractional Brownian motion in the theory of connectionless networks," *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 953–962, Aug. 1995.
[10] O. Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," in *Proc. 20th Annu. Conf. Local Computer Networks*, 1995, pp. 397–406.
[11] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*: Chapman & Hall, 1997.
[12] E. W. Knightly and N. B. Shroff, "Admission control for statistical QoS: Theory and practice," *IEEE Network*, pp. 20–29, Mar./Apr. 1999.
[13] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.
[14] J. Beran, R. Sherman, M. Taqqu, and W. Willinger, "Long-range dependence in variable-bit-rate video traffic," *IEEE Trans. Commun.*, vol. 43, pp. 1566–1579, 1995.
[15] M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. ACM SIGCOMM*, Aug. 1994, pp. 269–280.
[16] P. Abry and D. Veitch, "Wavelet analysis of long-range-dependent traffic," *IEEE Trans. Information Theory*, vol. 44, no. 1, pp. 2–15, 1998.
[17] V. O. K. Li, W. Liao, X. Qiu, and E. W. M. Wong, "Performance models of interactive video-on-demand systems," *IEEE J. Select. Areas Commun.*, vol. 14, no. 6, pp. 1099–1109, August 1996.
[18] L. Kleinrock, *Queueing Systems, Vol. I: Theory*: John Wiley & Sons, 1975.
[19] J. K. Dey-Sircar, J. D. Salehi, J. F. Kurose, and D. Towsley, "Providing VCR capabilities in large-scale video servers," *ACM Multimedia*, pp. 25–32, Oct. 1994.

Spiridon Bakiras received the B.S. degree in 1993 in electrical and computer engineering from the National Technical University of Athens, the M.S. degree in 1994 in telematics from the University of Surrey, and the Ph.D. degree in 2000 in electrical engineering from the University of Southern California. In May 2001 he joined the Department of Electrical and Electronic Engineering at the University of Hong Kong as a Research Assistant Professor. His research interests include multimedia communications, Internet QoS, web caching, and peer-to-peer systems.



Victor O. K. Li (F'92) was born in Hong Kong in 1954. He received the S.B., S.M., E.E., and Sc.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, in 1977, 1979, 1980, and 1981, respectively. He joined the University of Southern California (USC), Los Angeles, CA, USA in February 1981, and became Professor of Electrical Engineering and Director of the USC Communication Sciences Institute. Since September 1997 he has been with the University of Hong Kong, Hong Kong, where he is Chair Professor of Information Engineering and Managing Director of Versitech Ltd., the technology transfer and commercial arm of the University. He also serves on various corporate boards. His research is in information technology, including high-speed communication networks, wireless networks, and Internet technologies and applications. Sought by government, industry, and academic organizations, he has lectured and consulted extensively around the world. Professor Li is very active in the research community, and has chaired various international conferences and served on the editorial boards of various international journals. He has given distinguished lectures at universities around the world, and keynote speeches at many international conferences. Professor Li has received numerous awards, including, most recently, the Croucher Foundation Senior Research Fellowship, in March 2002, and the Bronze Bauhinia Star, Government of the Hong Kong Special Administrative Region, China, July 1, 2002. He was elected an IEEE Fellow in 1992.