Interactive Video Streaming with Proxy Servers

Martin Reisslein *
Arizona State Univ.
reisslein@asu.edu

Felix Hartanto
Chinese Univ. of Hong Kong
felix@ie.cuhk.edu.hk

Keith W. Ross
Institute Eurecom
ross@eurecom.fr

Received: June 2000
Revised: January 2001

Abstract

We study caching strategies for proxies that cache VBR–encoded continuous media objects for highly interactive streaming applications. First, we develop a model for streaming VBR–encoded continuous media objects. This model forms the basis for a stream admission control criterion and our study of caching strategies. We find that unlike conventional web caches, proxy caches for continuous media objects need to replicate or stripe objects to achieve high hit rates. We develop novel caching strategies that either implicitly or explicitly track the request pattern and cache (and replicate) objects accordingly. Our numerical results indicate that our caching strategies achieve significantly higher hit rates than caching without object replication.

Keywords: Caching, Continuous Media Object, Object Replication, Replacement Policy, Statistical QoS, Streaming, VBR Video.

1 Introduction

The dramatic growth of the World Wide Web has spurred the deployment of proxy caches. These store frequently requested objects close to the clients in the hope of satisfying future client requests without contacting the origin server. Highly localized request patterns, which exhibit hot–spots, i.e., frequent requests for a small number of popular objects, have made caching highly successful in reducing server load, network congestion, and client perceived latency. While most of the caching research to date has focused on caching of textual and image objects, web–based streaming of continuous media, such as video and audio, becomes increasingly popular. In fact, it is expected that by 2003, continuous media will account for more than 50 % of the data available on origin servers [13]. This trend is reflected in a recent study [1], which found that the number of continuous media objects stored on web servers has tripled in the first 9 months of 1998.

Caching and streaming of continuous media objects with proxy servers poses many new challenges [29]. These are due to the real–time constraints imposed by continuous media traffic and the high degree of interactivity expected by users. In this paper we focus on caching strategies for proxies that cache VBR–encoded continuous media for highly interactive streaming applications in disk arrays. We consider an architecture where the proxy servers cache frequently requested continuous media objects in their local storage, which is typically a disk array. The clients direct their streaming requests to their assigned proxy server. If the proxy can satisfy the streaming request — a cache hit — the object is streamed from the proxy to the client. If the proxy can not satisfy the request — a cache miss — the object is obtained from the appropriate origin server and the proxy decides according to a caching strategy whether or not to cache the object.

The contribution of this paper is twofold. First, we develop a stream model for streaming VBR–encoded continuous media objects from the proxy's disk array over an access network to the clients. Based on the stream model we design a scheme for admission control and resource reservation that provides stringent statistical Quality of Service (QoS) guarantees.

Our second contribution is to study caching strategies for continuous media objects. Our study shows that unlike conventional web caches, proxy caches for continuous media should replicate or stripe objects to accommodate the typically highly localized request patterns and to ensure good stream quality. We develop natural extensions of conventional caching strategies which *implicitly track* the client request pattern by combining object replication with a conventional replacement policy, such as LRU or LFU. We then develop and evaluate a novel caching strategy which *explicitly tracks* the client request pattern and caches objects accordingly. Our numerical results indicate that the hit rate achieved by our caching strategy with explicit tracking is almost 20 % higher than the hit rate

*Corresponding Author: Martin Reisslein, Dept. of Electrical Eng., Arizona State University, P.O. Box 877206, Tempe AZ 85287–7206, phone: (480)965–8593, fax: (480)965–8325, reisslein@asu.edu, http://www.eas.asu.edu/~mre.

of caching with implicit tracking. Caching with implicit tracking in turn achieves typically 10 % higher hit rates than conventional caching without object replication.

1.1 Related Work

There are only few studies of caching and streaming of continuous media objects with proxy servers which are complementary to the issues studied in this paper. Rejaie *et al.* propose a proxy caching mechanism [26] in conjunction with a congestion control mechanism [24, 25] for layered–encoded video. With layered encoding the compressed video stream is split into a base layer, which contains low quality encoding information, and enhancement layers, which improve the stream quality. The basic idea of their caching mechanism is to cache layers according to the objects' popularities: the more popular an object, the more layers are cached. When streaming an object to a client, the layers that are not cached at the proxy are obtained from the origin server. A related idea is explored by Wang *et al.* in their study on video staging [39]. With video staging the part of the VBR video stream, that exceeds a certain cut–off rate (i.e., the bursts of the VBR stream) is cached at the proxy while the lower (now smoother) part of the video stream is stored at the origin server. Our work is complementary to these studies on caching of video layers. Our focus is on (1) developing a stream model and admission control conditions that ensure statistical QoS for continuous media streaming, and (2) object replication and striping to accommodate the typically highly localized client request pattern while providing good stream quality.

Sen *et al.* [33] propose to cache a prefix (i.e., the initial frames) of video streams at the proxy and to employ work–ahead smoothing while streaming the object from the proxy to the client. The cached prefix hides the potentially large initial start–up delay of the work–ahead transmission schedule from the client. A major drawback of this approach is that it is not suited for interactive video streaming. The client experiences a potentially large delay after invoking an interaction (such as a temporal jump) since the work–ahead smoothing schedule has to build up a buffered reserve at the client before playback can resume.

Tewari *et al.* [38] propose a Resource Based Caching (RBC) scheme for Constant Bit Rate (CBR) encoded video objects. They model the cache as a two resource (storage space and bandwidth) constrained knapsack and study replacement policies that take the objects' sizes as well as CBR bandwidth into account. Our work differs from RBC in that we consider VBR encoded video objects. Also, object replication and striping as well as interactive streaming are not addressed by Tewari *et al.*

There is a large body of literature on striping of video objects in the context of video servers. Most of these studies assume that the videos are CBR encoded; see for instance [12, 8, 19]. Striping for VBR encoded video objects is studied by Shenoy and Vin [35]. They develop an analytical model for the most heavily loaded disk and study the optimal striping placement. Sahu *et al.* [30] study round based retrieval strategies for VBR video from disk. These studies on striping and retrieval of VBR video assume that the user request pattern is uniform and do not consider interactive delays.

Birk [2] proposed an approach where the video blocks are placed randomly on the disk array to overcome the hot–spot problem. In his scheme interactive requests, which result from client interactions, are given priority over sequential retrievals to ensure short interactive delays. This approach appears promising in the context of proxy streaming of interactive VBR video, although there are some issues that require further research. Most importantly, a stream admission control rule that ensures statistical QoS when retrieving randomly placed blocks of VBR video from the disk array requires more research. Also, the performance of the scheme when the proportion of interactive requests is high needs to be examined.

2 Architecture

In this section we describe the end–to–end architecture for the delivery of continuous media objects using proxy servers. The architecture is illustrated in Figure 1. The

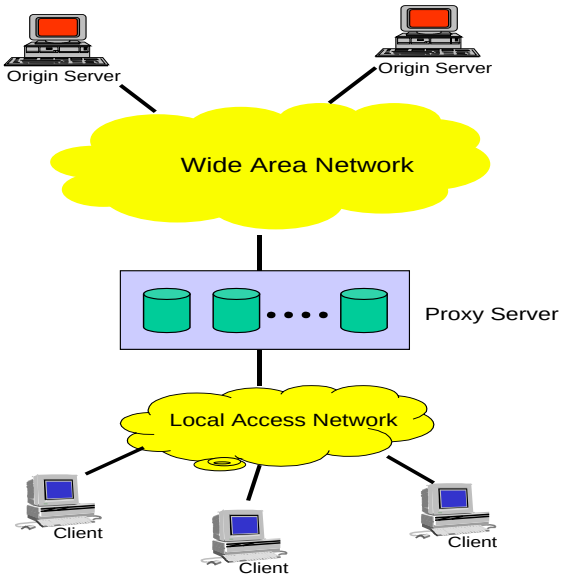


Figure 1: Architecture for continuous media streaming with proxy server.

continuous media objects are stored on the origin servers. The continuous media objects are prerecorded audio and video objects, such as CD–quality music clips, short video clips (e.g., trailers or music videos) or full–length movies

or on–line lectures. The proxy server is located close to the clients. It is connected to the origin servers via a wide area network (e.g., the Internet). The proxy server is connected to the clients via a local access network. The local access network could be a LAN running over Ethernet, or a residential access network using xDSL or HFC technologies.

In the following we briefly outline the delivery procedure for continuous media objects. The client directs its request for a particular continuous media object to its assigned proxy server (for instance by using the Real Time Streaming Protocol (RTSP) [32]). If the continuous media object is not cached in the proxy, that is, in the case of a cache miss, the proxy forwards the request to the appropriate origin server. The origin server streams the continuous media object to the proxy server. The proxy relays the stream to the requesting client and at the same time caches the continuous media stream in its local storage. If the local storage (typically disk array) at the proxy is full the proxy decides according to a replacement policy (see Section 5) which continuous media object to remove from the cache to make room for the new object. If the replacement algorithm fails to free up enough disk space for the the new objects (this is the case when not enough objects can be removed without interrupting ongoing streams; see Section 5) the object is streamed from the origin server directly to the client. In the case of a cache miss the proxy server does not reduce the bandwidth usage in the wide area network, neither does it improve the stream quality and the level of interactivity offered to the client.

In the case of a cache hit, that is, when the continuous media object requested by the client is cached in the proxy's disk array, the object is streamed from the proxy over the local access network to the client.

Before the streaming commences the proxy conducts in both cases admission control tests to verify whether the available disk bandwidth and the bandwidth in the local access network are sufficient to support the new stream. Only if the admission tests are successful is the requested object streamed from the origin server (in the case of a cache miss) or from the proxy (in the case of a cache hit).

3 Model for Continuous Media Streaming from Proxy

In this section we model the streaming of continuous media from the proxy. Our analysis applies to any type of continuous media traffic, however, to fix ideas we focus on streaming of video objects. We naturally assume that the video objects are Variable Bit Rate (VBR) encoded. For VBR encoding the quantization scale is kept constant to maintain high video quality even for high action scenes. For the same quality level the file size and average bit rate of a Constant Bit Rate (CBR) encoded movie or sports clip are typically two times or more than the file size and average bit rate of the VBR encoding [4, 37]. Our first contribution is to develop a unified scheme for admission control and resource reservation in the proxy server as well as the local access network. Toward this end we first develop a *disk model* and derive the effective disk bandwidth for the retrieval of continuous media traffic with tight interactive delay constraints. We then develop a *stream model* for the VBR–encoded continuous media traffic and design a scheme for admission control and resource reservation that provides stringent statistical QoS.

3.1 Disk Model

We assume that each disk in the proxy's disk array consists of single platter side and a single arm. We assume that the proxy server retrieves data for the ongoing video streams in constant–time rounds; we denote the round length by T. We also assume that each disk in the disk array uses the SCAN scheduling algorithm [20]. Specifically, in each round, each disk arm sweeps across its entire platter exactly once with no back tracking. With the SCAN scheduling algorithm, the overhead incurred within a round for a given disk is

$$\text{disk overhead} = l_{\text{seek}} + Il_{\text{rot}},$$

where I is the number of streams that the disk is servicing. The constant l_{seek} is the maximum seek time of the disk (i.e., the time to move the arm from the center to the edge of the platter, which is equal to the time to move the arm from the edge to the center of the platter). The constant l_{rot} is the per–stream latency, which includes the maximum rotation time of the disk and the track–to–track seek time. Table 1 summarizes our disk notation and the nominal values for the disk parameters. The nominal parameters reflect the current performance of high–speed disks [36].

Table 1: Nominal values of disk parameters

Parameters	Notation	Nominal value
disk size	X	13 Gbytes
disk transfer rate	r	8.5 Mbytes/sec
maximum seek time	l_{seek}	18 msec
rotational latency	l_{rot}	5 msec

The initial start–up delay as well as the responsiveness to an interactive request (pause/resume or a temporal jump) is typically modeled to be twice the round length, $2T$, when the SCAN algorithm is used. This delay model is based on the worst–case assumption that the request of the user arrives just after the start of a round, say round k, and arrives too late to be scheduled by the SCAN algorithm for round k. The request has to wait for the start of the next round. The request is included in the disk read schedule of round $k+1$ and the requested video data is read into

the disk buffer during round $k+1$. The disk buffer of round $k+1$ becomes the network buffer of round $k+2$ and the transmission of the requested video data out of the network buffer starts at the beginning of round $k+2$. Thus, the disk–subsystem introduces a maximum delay of two rounds, i.e., $2T$. We shall assume that the maximum disk–subsystem delay is constrained to 0.5 sec. Therefore, we use a round length of $T = 0.25$ sec. Note that the total interactive delay also includes transmission delays as well as client de–smoothing and decoding delays. These additional delays add another 0.25 sec to 0.5 sec to the 0.5 sec disk–subsystem delay, giving a total delay on the order of .75 sec to 1.0 sec. Thus, with a round length of 0.25 sec the system is able to give the user a pleasurable interactive experience with less than 1 second delay for all interactions.

For the development of the disk model we assume for now that the video objects are placed in the proxy's disk array using the *localized placement strategy*. With the localized placement strategy each video file is placed contiguously on a single disk. We shall later study a number of more complex *striping placement strategies*, whereby each video file is striped across a subset of the disks in the proxy's disk array.

Now consider one of the disks in the proxy's disk array and suppose that this disk is servicing I streams. Let $retr(I,T)$ denote the number of bits retrieved for the I streams in one round of length T. The disk transfers this video data at the disk transfer rate r. Thus the total disk transfer time within a round is $retr(I,T)/r$. The total disk overhead within a round is $l_{\text{seek}} + Il_{\text{rot}}$. Thus the amount of time the disk requires to service the I ongoing streams in a round is $retr(I,T)/r + l_{\text{seek}} + Il_{\text{rot}}$. For lossless service the time required to service the I streams in a round must be no greater than the round length itself:

$$\frac{retr(I,T)}{r} + l_{\text{seek}} + Il_{\text{rot}} + \leq T.$$

Rearranging the terms in the above inequality, we obtain the maximum streaming rate for lossless service:

$$\frac{retr(I,T)}{T} \leq r\left(1 - \frac{l_{\text{seek}} + Il_{\text{rot}}}{T}\right) =: C_{\text{disk}}, \qquad (1)$$

which we refer to as *disk bandwidth*. With the disk parameters from Table 1 the disk bandwidth is $(63.1 - 1.36 \cdot I)$ Mbps. The disk bandwidth is obviously upper bounded by the disk transfer rate. Note that the disk bandwidth increases for increasing round length T. We therefore use a round length of $T = 0.25$ sec, the largest round length that guarantees a maximum interactive delay of 1 second. Note furthermore that the disk bandwidth decreases as the number of ongoing streams I increases. This is because the disk wastes a larger fraction of the round with seeks and rotations when the number of ongoing streams increases.

3.2 Stream Model

We now develop a model for the VBR video streams that are retrieved from the proxy's disk array and sent over the local access network to the clients. We assume in the basic stream model that the video objects are retrieved from a single disk, that is, we assume localized placement in the proxy's disk array. We shall consider streaming from a disk array with striping placement later.

In our stream model we assume random stream phases, which accurately model interactive streaming. We base the stream model on the distribution of the frame sizes, as proposed in [21]. We chose this approach because it provides simple and accurate admission control decisions through the Large Deviation Approximation [27]. The many models that are based on Markov modulated processes (e.g., [18]) model traffic quite accurately. However, for admission control they are either more complex (requiring the calculation of many state transition probabilities) or employ the asymptotic theory of effective bandwidth which is less accurate for small buffers and bursty video traffic [15].

Consider a single disk in the proxy's disk array and suppose that this disk is streaming I video objects. For simplicity we assume that each video object has N frames and a frame rate of F frames per second. Let $f_n(i)$ denote the number of bits in the nth encoded frame of video object i, $i = 1, \ldots, I$. We assume that all video objects are cached in the proxy; the *frame size trace* $\{f_n(i),\ 1 \leq n \leq N\}$ for video object i is therefore a sequence of known integers. As pointed out above the video frames are retrieved from disk in rounds of length T. For each ongoing video stream let K denote the number of frames retrieved in one round; clearly $K = T \cdot F$. (In our numerical work we use a round length of $T = 0.25$ sec and a frame rate of $F = 24$ frames/sec, thus $K = 6$ in our numerical examples.) Following the terminology of the file server literature [34] we refer to the K frames retrieved in one round as *block*. Let $x_m(i)$ denote the size of the block (in bits) retrieved for video stream i in round m. Assuming that the frames of the video object are retrieved strictly sequentially, that is, the first K frames are retrieved in round 1, the next K frames are retrieved in round 2, and so on; in other words by excluding client interactions, we have

$$x_m(i) = \sum_{n=(m-1)K+1}^{mK} f_n(i), \qquad m = 1, \ldots, \frac{N}{K}.$$

We refer to the sequence $\{x_m(i),\ 1 \leq m \leq N/K\}$ as *block size trace* for stream i. Following [21] we model the random start times of the video streams and the client interactions by assigning to video object i the random phase θ_i. (These client interactions such as pause/resume and forward/backward temporal jumps can be communicated to the proxy using the Real Time Streaming Protocol (RTSP) [32]; we assume in our model that the temporal jumps have

the granularity of blocks, i.e., K frames.) It is natural to assume that the random phases $\theta_i,\ i = 1,\ldots,I$, are independent and uniformly distributed over $[0,\ N-1]$. In our model the amount of data retrieved from disk for video stream i in round m is

$$X_m(i) = x_{m+\theta_i}(i),$$

where the index $m+\theta_i$ is wrapped around if it exceeds the number of blocks N/K of the video object. The total amount of data retrieved in round m for the I ongoing video streams is

$$X_m = \sum_{i=1}^{I} X_m(i) = \sum_{i=1}^{I} x_{m+\theta_i}(i). \tag{2}$$

We now briefly summarize the main implications of our stream model:

1. For each fixed round index m, $X_m(1),\ldots,X_m(I)$ are independent random variables.

2. The probability mass function of $X_m(i)$ can be obtained directly from the block size trace of the cached video object:

$$P(X_m(i) = l) = \frac{K}{N}\sum_{m=1}^{N/K} 1(x_m(i) = l).$$

   Note that the distribution of $X_m(i)$ does not depend on the round index m. To simplify notation we write henceforth $X(i)$ for $X_m(i)$ and X for X_m.

We now proceed to develop stream admission rules that ensure a high user perceived quality of the streamed continuous media while efficiently utilizing the bandwidth resources in the proxy's disk array and the local access network. Toward this end we define *statistical* QoS requirements. Specifically, we define the loss probability as the long run average fraction of information (bits) lost due to the limited bandwidth (in disk array and local access network) and admit a new stream only if the loss is less than some miniscule ϵ, such as $\epsilon = 10^{-6}$. Formally, the loss probability due to the limited disk bandwidth is given by

$$P_{\text{loss}}^{\text{disk}} = \frac{E[(X - C_{\text{disk}}T)^+]}{E[X]}, \tag{3}$$

where the expectation is over all possible phase profiles. Note that up to this point we have considered a single disk in the proxy's disk array. To formally define the loss probability due to the limited bandwidth in the local access network we consider the aggregate streaming rate from the proxy's disk array (resulting from cache hits) as well as the streaming from the origin servers (resulting from cache misses). Let D denote the number of disks in the proxy's disk array and let X^d be the random variable denoting the amount of data retrieved in one round from disk $d,\ d = 1,\ldots,D$. The aggregate amount of data retrieved from the proxy's D disks in one round is $\sum_{d=1}^{D} X^d$. Furthermore, let X^o be the random variable denoting the amount of data fed into the local access network in one round from the origin servers. The total amount of data fed into the local access network in one round is

$$Y = \sum_{d=1}^{D} X^d + X^o. \tag{4}$$

The network loss probability is

$$P_{\text{loss}}^{\text{net}} = \frac{E[(Y - C_{\text{net}}T)^+]}{E[Y]}, \tag{5}$$

where C_{net} denotes the bandwidth available for streaming continuous media objects into the local access network. This bandwidth could, for instance, be the bandwidth of the link connecting the proxy to an xDSL central office, or the bandwidth of the cable trunk that the proxy feeds into.

The overall streaming loss probability is bounded by the sum of the disk and network loss probabilities. Our statistical QoS requirement is that the streaming loss probability be less than some miniscule ϵ:

$$P_{\text{loss}}^{\text{disk}} + P_{\text{loss}}^{\text{net}} \le \epsilon. \tag{6}$$

Before granting a new streaming request we verify whether (6) continues to hold when including the new stream in (2) (for the appropriate disk; recall we are assuming localized placement) and (4).

Evaluating the probabilities in (6) involves the convolution of independent random variables, which often leads to numerical problems. We therefore apply the Large Deviation approximation, which is known to be accurate and computationally efficient [27]. Let $\mu_X(s)$ denote the logarithmic moment generating function of X, the amount of data retrieved from a given disk in one round,

$$\mu_X(s) = \ln E[e^{sX}].$$

Clearly,

$$\mu_X(s) = \sum_{i=1}^{I} \mu_{X(i)}(s),$$

by the independence of the $X(i)$'s. The individual $\mu_{X(i)}(s)$'s are easily obtained from the respective round size traces. The Large Deviation approximation for the disk loss probability is [27]:

$$P_{\text{loss}}^{\text{disk}} \approx \frac{1}{E[X]s^{\star 2}\sqrt{2\pi\mu_X''(s^\star)}} e^{-s^\star C_{\text{disk}}T + \mu_X(s^\star)}, \tag{7}$$

where $s^\star$ satisfies

$$\mu'_X(s^\star) = C_{\text{disk}}T.$$

Assuming that the streams retrieved from the D disks in the proxy's disk array are mutually independent it is straightforward to write out the corresponding Large Deviation approximation for $P_{\text{loss}}^{\text{net}}$.

3.3 Striping Placement

In this section we study streaming from a proxy that uses striping placement of video objects in its disk array. Recall that D denotes the number of disks in the proxy's disk array. We shall at first focus on *full striping*, whereby each video object is striped over all D disks in the proxy. There are essentially two different striping techniques: Fine Grained Striping (FGS) and Coarse Grained Striping (CGS) [7, 17]. With Fine Grained Striping each block (consisting of K frames) is segmented into D equal–sized parts, called stripes, and each of the disks stores one of the block's stripes. When retrieving a block from the disk array, the server reads all D stripes in parallel. With Coarse Grained Striping (also referred to as Data Interleaving in [11]) each block is stored on a separate disk. The blocks are typically assigned to the disks in a round robin manner, When the proxy retrieves a block from its disk array it reads the entire block from one disk. Therefore, CGS has less overhead than FGS (since the proxy has to access D disks to retrieve one block with FGS). The drawback of CGS, however, is its large interactive delay, which is due to the large scheduling delay for disk accesses in disk arrays with CGS [7, 17, 23]. The large scheduling delay with CGS severely limits the number of streams that a disk array with CGS can support when a tight interactive delay constraint is imposed. In fact, it is shown in [23] that given a tight interactive delay constraint of 1 second CGS typically supports fewer streams than FGS. We are interested in continuous media streaming with a high degree of interactivity and focus therefore on FGS in this paper.

We now proceed to develop a model for streaming from a disk array with FGS placement. For that purpose we adapt the disk model (Section 3.1) and stream model (Section 3.2) for localized placement. First, we consider the disk model. Suppose that the proxy's disk array consists of D disks. Suppose that the proxy is servicing J streams. Consider one of the D disks. The disk will transfer J stripes in one round. With J disk accesses the disk overhead incurred in one round is

$$\text{disk overhead} = l_{\text{seek}} + Jl_{\text{rot}}.$$

With a derivation that parallels the development of the disk model for localized placement in Section 3.1 we obtain for the disk bandwidth with FGS:

$$C_{\text{disk}}^{\text{FGS}} = r\left(1 - \frac{l_{\text{seek}} + Jl_{\text{rot}}}{T}\right).$$

We now adapt the stream model of Section 3.2 to FGS. Consider again one of the D disks. Let $X^{\text{FGS}}(j)$ be the random variable denoting the amount of data (i.e., the size of the stripe in bits) retrieved for stream j, $j = 1, \ldots, J$, in a given round. Recall that the stripes are obtained by dividing each block of a video object into D equal–sized segments. With our stream model the probability mass function of $X^{\text{FGS}}(j)$ can thus be directly obtained from the block size trace of the cached video object:

$$P(X^{\text{FGS}}(j) = l) = \frac{K}{N}\sum_{m=1}^{N/K} 1(\frac{x_m(j)}{D} = l).$$

The total amount of data retrieved from the disk in a given round is

$$X^{\text{FGS}} = \sum_{j=1}^{J} X^{\text{FGS}}(j),$$

and the aggregate amount of data retrieved from the entire disk array is $Y^{\text{FGS}} = DX^{\text{FGS}}$. It is now straightforward to compute the loss probabilities $P_{\text{loss}}^{\text{disk}}$ and $P_{\text{loss}}^{\text{net}}$ using the Large Deviation approximation.

We finally consider *group striping*. With group striping the video objects are striped over $W \leq D$ disks. We refer to W as the striping width. Localized placement ($W = 1$) and full striping ($W = D$) are special cases of group striping. With group striping the proxy's disk array is typically segmented into striping groups, which consist of W disks each. Each cached video object is striped within a striping group. With FGS each block of a video object is segmented into W equal–sized stripes and each disk in the striping group stores one of the block's stripes. The disk model and stream model for streaming from a proxy with group striping are straightforward extensions of the models for full striping.

4 Replication and Striping of Video Objects

In this section we study the impact of the placement of video objects in the proxy's disk array on the proxy's performance. We show that replication and striping of popular objects in the proxy significantly improve the hit rate and throughput of the proxy as well as the user–perceived media quality.

Throughout our performance study we assume that the requests for continuous media objects follow the Zipf distribution [40]. Specifically, if there are M objects, with object 1 being the most popular and object M being the least popular, then the probability that the mth most popular object is requested is

$$q_m = K/m^\zeta, \qquad m = 1, \ldots, M,$$

where

$$K = \frac{1}{1 + 1/2^{\zeta} + \cdots + 1/M^{\zeta}}.$$

The Zipf distribution, which is characterized by the parameter $\zeta \geq 0$, corresponds to a highly localized request pattern. It has been observed that the requests for movies in video rental stores and Video on Demand systems follow a Zipf distribution with ζ around one [5]. Furthermore, studies of web caches have shown that requests for images and HTML documents are well described by a Zipf distribution with a ζ of roughly one [3]. We expect therefore that requests for on–line continuous media objects will also follow a Zipf distribution.

For the numerical investigation in this paper we use traces of MPEG encoded video objects. We obtained 7 MPEG–1 traces, which give the number of bits in each encoded video frame, from the public domain [10, 16, 28]. The 7 video traces were used to create 10 pseudo traces each 40,000 frames long. The statistics of the resulting traces are summarized in Table 2. The *bean, bond, lambs,*

Table 2: Trace statistics

	Frames		
Trace	Peak [Mbit/s]	Peak/Mean	Std. Dev.
bean	24.9	13.0	2.25
bond	19.3	10.1	2.03
lambs	35.2	18.3	2.94
oz	16.1	8.4	2.39
soccer	13.2	6.9	1.84
star wars 1	20.9	10.9	2.35
star wars 2	25.3	13.2	2.25
star wars 3	23.0	12.0	2.22
star wars 4	16.2	8.4	2.05
terminator	14.0	7.3	1.79

soccer, and *terminator* traces were created by multiplying the frame sizes of the video traces from [28] by a constant to bring their average bit rates to 2 Mbps. The *oz* trace was created by taking the first 40,000 frames of the MPEG encoding from [16] and multiplying the frame sizes by a constant to raise the average bit rate to 2 Mbps. Finally, the four *star wars* traces were obtained by dividing the MPEG encoding from [10] into four segments of 40,000 frames each and then raising the average bit rate of the segments to 2 Mbps. Although the ten pseudo traces are not traces of actual videos objects, we believe that they reflect the characteristics of MPEG–2 encoded video objects (highly bursty, long–range scene dependence, average rate about 2 Mbps). In summary, we have 10 VBR encoded video objects with $N = 40{,}000$ frames and a frame rate of $F = 24$ frames/sec.

In our performance evaluation we focus on the impact of the object placement and caching strategies in the proxy's disk array on the proxy performance. Specifically, we investigate the object placement and caching strategies that utilize the storage capacity and disk bandwidth of the proxy's disk array most efficiently. To highlight the impact of the object placement and caching strategies we do not include the streaming over the local access network in our study, that is, we focus on the admission control condition $P_{\text{loss}}^{\text{disk}} \leq \epsilon$. We refer the interested reader to [22, 31, 14, 6] for studies of continuous media streaming over local access networks.

To motivate the replication and striping of video objects in the proxy's disk array, we first consider a very simple caching scenario. Suppose that the 10 video objects from Table 2 are cached in the proxy's disk array. Furthermore, suppose, that each video object is placed on its own disk, that is, a localized placement strategy with one video object per disk is employed. We use the disk model and streaming model of Section 3 to evaluate this simple caching scenario. We impose the statistical QoS requirement that the long run average fraction of video information (bits) lost due to the limited disk bandwidth be less than 10^{-6}, i.e. $P_{\text{loss}}^{\text{disk}} \leq 10^{-6}$. For each video object we use the large deviation approximation (7) to calculate the maximum number of simultaneous streams that can be supported by a single disk. The results are reported in Table 3. The table also gives the maximum number of simultaneous streams that can be supported when peak rate allocation is used. The video objects have an average rate of 2 Mbps, thus the stability limit is 19 streams. We observe from the table that the statistical admission control criterion allows for significantly more streams than peak rate allocation. This substantial multiplexing gain comes at the expense of small loss probabilities of the order of 10^{-6}. These miniscule losses, however, can be effectively hidden by error concealment techniques and will therefore not be noticed by the viewers. We also observe from Table 3 that the number of simultaneous streams supported by a disk depends on the burstiness of the stored video object. The disk with the *lambs* video object, which has the largest peak–to–mean ratio, supports the smallest number of simultaneous streams. On the other hand, the disks stor-

Table 3: Number of streams that can be supported by a single disk.

Trace	Stat. Mux.	Peak Allocation
bean	12	2
bond	15	3
lambs	11	1
oz	14	3
soccer	15	4
star wars 1	14	2
star wars 2	14	2
star wars 3	14	2
star wars 4	14	3
terminator	15	4

ing the *soccer* and *terminator* video objects, which have the smallest peak–to–mean ratio, support the most streams.

Next, we study the total number of streams that the proxy can typically support, when the requests for the 10 video objects are distributed according to a Zipf distribution with $\zeta = 1$. (We still assume localized placement with one video object per disk, that is, there is one disk with *bean*, one disk with *bond*, and so on.) For this illustrative example we assume that the popularity of the video objects in Table 2 decreases in alphabetical order, that is, *bean* is the most popular object (requested with probability $q_1 = 0.341$) and *terminator* is the least popular object (requested with the probability $q_{10} = 0.034$). We determinate the typical number of streams, that the proxy can simultaneously support with the following procedure. For a given target number of streams S we generate S requests from the Zipf distribution. We then determine the number of requests that can be supported by the 10 disks using the results from Table 3. We repeat the experiment 1000 times, creating $1000 \cdot S$ requests. If 95 % of these requests can be supported, then we increment S and repeat the entire procedure. The procedure continues until the 95 % criterion is violated. We find with this procedure that the proxy can typically support 39 simultaneous streams. This, however, is only a small fraction of the disk array's capacity of 138 simultaneous streams (found by adding up the "Stat. Mux" column of Table 3).

The reason for this is twofold. First, due to the limited disk bandwidth the proxy cannot satisfy many of the requests for the most popular objects. Secondly, much of the disk bandwidth of the disks housing the less popular objects is underutilized. This phenomenon is commonly referred to as *hot–spot problem*. The hot–spot problem severely affects the proxy's performance. The proxy either has to reject many requests for the most popular objects (and the clients have to obtain the objects directly from the origin server) or it has to compromise the stream quality by admitting more streams than the QoS criterion $P_{\text{loss}}^{\text{disk}} \leq 10^{-6}$ allows. Both of these options, however, are highly undesirable, as they increase the load on the wide area network and reduce the media quality and level of interactivity offered to the clients. We are thus motivated to study strategies that overcome the hot–spot problem by utilizing the proxy's storage capacity and disk bandwidth more efficiently. Specifically, we study two strategies:

- Object replication: The proxy stores more than one copy of the popular video objects. The goal is to overcome the hot–spot problem by roughly matching the replication distribution (i.e., the distribution of the number of copies of the objects) to the request distribution.

- Striping placement: The video objects are striped over a subset of the disks in the proxy's disk array. This allows the proxy to use the aggregate disk bandwidth of the entire subset to stream the video objects. If the video objects are striped over the entire disk array (full striping) then the hot–spot problem vanishes and all request distributions can be equally accommodated.

Recall that streaming from a proxy with striping placement has been discussed in Section 3.3.

We now proceed to discuss object replication in detail. To simplify the discussion we initially assume localized placement. (We shall later study object replication in conjunction with striping). To make the idea of object replication a little more precise, let D denote the number of disks in the proxy's disk array. Let M denote the number of distinct objects in the proxy's cache. Let $C_m,\ m = 1, \ldots, M$, denote the number of copies of object m in the cache. For simplicity, we initially assume that each disk stores exactly one video object, thus $\sum_{m=1}^{M} C_m = D$. Now suppose that the request pattern for the M object has a known distribution (perhaps a Zipf distribution with known parameter ζ). To make the replication distribution approximately equal to the request distribution we replicate the objects according to the following *replication algorithm*:

1. $C_m = \lfloor q_m D \rfloor,\ m = 1, \ldots, M$.
2. If $C_m = 0$, set $C_m = 1$.
3. Calculate $C = C_1 + \cdots + C_M$.
4. If $C > D$, decrement C_m for the least popular object with $C_m > 1$, then for the next least popular object with $C_m > 1$, and so on, until $C = D$.
5. If $C < D$, increment C_m for the most popular object, then for the next most popular object, and so on, until $C = D$.

Algorithm 1: Non-uniform replication algorithm.

This concludes our discussion of object replication for localized placement. The concept extends to group striping with striping width $W > 1$ in a straightforward manner. With group striping the video objects are replicated on distinct striping groups.

We have conducted a numerical study of object replication and striping placement. In the numerical study we consider a proxy with a disk array consisting of $D = 100$ disks. We use the $M = 10$ video objects from Table 2. In the numerical study the requests for the video objects follow a Zipf distribution with $\zeta = 1$. We use the replication algorithm (see Algorithm 1) to match the number of copies of the video objects to the request distribution. We then use the 95 % criterion to determine the number of simultaneous streams that the proxy can typically support. The

results are reported in Table 4 for different striping widths W.

Table 4: Number of simultaneous streams that the proxy can typically support for different replication strategies and striping widths.

	replication strategy		
W	uniform	request	request + bw
1	390	1274	1303
2	622	1044	1045
5	562	633	632
10	380	380	380

The table also gives the number of simultaneous streams that can typically be supported when the video objects are uniformly replicated, that is, there are $D/M = 10$ copies of each video object in the disk array.

Two points are especially noteworthy. First, we observe that replicating objects according to the clients' request pattern significantly increases the number of streams that the proxy can typically support. For localized placement ($W = 1$) the streaming capacity of the proxy is increased roughly threefold by taking the request pattern into account. The second noteworthy observation is that for uniform replication the streaming capacity increases as the striping width increases from one to two. This is because striping over two disks alleviates the hot–spot problem and thus allows the proxy to better accommodate the clients' requests. As the striping width is increased further, however, the increased seek and rotational overhead of striping becomes the dominant effect, reducing the streaming capacity of the proxy. For the proxy with object replication according to the request pattern localized placement (W =1) gives the largest streaming capacity. This is because localized placement minimizes the disk overhead while object replication according to the request pattern overcomes the hot–spot problem.

Table 4 also gives the number of simultaneous streams that the proxy can typically support when the object replication takes the video objects' popularity as well as bandwidth demand into account. This approach is motivated by the results from Table 3, which indicate that disks housing relatively bursty video objects can support relatively fewer simultaneous streams. To accommodate a given request pattern the proxy should therefore house more copies of objects that consume relatively more disk bandwidth. To make this idea a little more precise let b_m, $m = 1, \ldots, M$, denote the maximum number of simultaneous streams of object m that can be supported by a single disk. (For the video objects used in the numerical study the b_m's are given in Table 3.) To take an object's bandwidth demand into account we set

$$C_m = \lfloor q_m D \frac{1}{b_m M} \sum_{l=1}^{M} b_l \rfloor$$

in Step 1 of Algorithm 1. The factor $\frac{1}{b_m M} \sum_{l=1}^{M} b_l$ is larger (smaller) than one for video objects that require relatively more (less) bandwidth. We see from Table 4 that taking the objects' bandwidth demand into consideration increases the proxy performance slightly for localized placement ($W = 1$). For striping placement this replication strategy does not improve the proxy performance.

We next study the robustness of the replication and striping strategies of Table 4 with respect to changes in the request pattern. For this purpose we vary the parameter of the Zipf distribution from which the requests are generated. Throughout this experiment the video objects are replicated according to a Zipf distribution with fixed parameter $\zeta = 1$ (that is, throughout we use the object replication used in the previous experiment). In other words, the request distribution varies while the replication distribution is held fixed. Figure 2 shows the typical number of simultaneous streams that the proxy can support as a function of the Zipf parameter of the request distribution. The figure gives plots for uniform replication and replication according to Zipf distribution with $\zeta = 1$ for localized placement and striping over two disks. We see

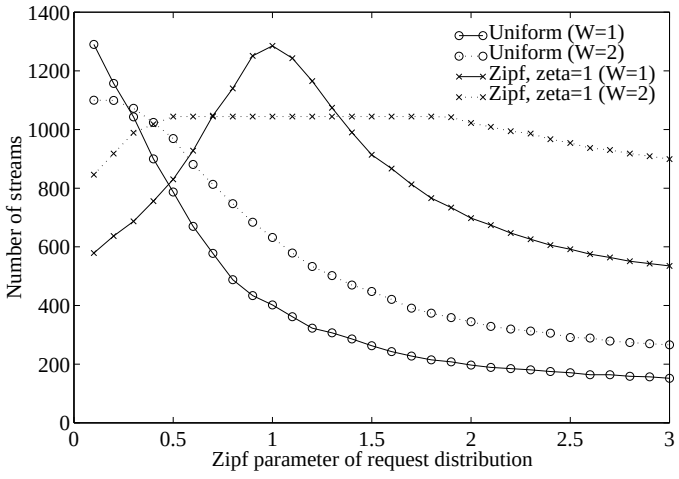


Figure 2: Robustness of replication strategies.

from the figure that uniform object replication gives good performance only when the client request pattern is fairly uniform, that is, when the Zipf coefficient of the request pattern is small. For the skewed request distributions that are typical for client request patterns, uniform replication even with striping gives poor proxy performance. Striping over two disks with object replication according to the

Zipf distribution with $\zeta = 1$ — which can be thought of as an estimate of the client request pattern — is very robust to changes in the client request pattern. This strategy can support close to 1050 streams over a wide range of the Zipf coefficient of the actual request distribution. Striping placement with object replication according to an estimate of the client request pattern thus performs well even when this estimate differs significantly from the actual request pattern. However, with a good estimate of the request pattern, localized placement ($W = 1$) with object replication according the estimate outperforms striping placement. In the studied example, where the estimate of the request pattern is the Zipf distribution with $\zeta = 1$, localized placement with object replication according to this estimate outperforms striping placement when the Zipf parameter of the actual request pattern is between 0.6 and 1.4.

The localized placement strategy has the added advantage that it is very simple and works well for heterogeneous disk arrays consisting of disks with different performance characteristics. Furthermore, localized placement avoids the availability problem of striping placement — if one disk fails then all video files that are striped over this disk become unavailable to the clients. (With localized placement a given disk stores parts of fewer video objects therefore disk failure has less impact of availability.) The availability problem of striping can be mitigated (at the expense of added complexity) through mirroring of video blocks or storing of parity information of the video blocks; see for instance [9] and references therein. Because of its simplicity and potential for improved performance we focus on localized placement ($W = 1$) in the next section on caching strategies, however, the studied caching strategies apply equally well to striping placement.

5 Caching Strategies

In the previous section, which served to motivate object replication and striping in the proxy, we assumed that (*i*) the requests for video objects follow a known distribution, and (*ii*) all available objects are cached in the proxy. In this section we consider a more realistic scenario, where (*i*) the client request pattern is unknown, and (*ii*) only a subset of the available objects can be cached in the proxy. We propose and evaluate caching and replacement policies that either implicitly or explicitly track the client request pattern. The *caching policy* determines which object (and how many copies thereof) to cache while the *replacement policy* determines which objects to evict from the cache to free up storage space for new objects.

5.1 Implicit Tracking

With implicit tracking the caching policy is invoked whenever the proxy can not satisfy a client's streaming request. This is the case when either (1) the requested object is not cached, or (2) the requested object is cached but the additional stream can not be supported by the cached copy without violating the QoS requirement $P_{\text{loss}}^{\text{disk}} \leq \epsilon$. The basic caching policy is to always try to cache the requested object in case (1). In case (2) we distinguish two policies: *caching with object replication* and *caching without object replication*. Caching with object replication attempts to cache an additional copy of the requested object (which is generated internally from the already cached copy). Caching without object replication, on the other hand, leaves the cache contents unchanged and the requested object is streamed from the origin server directly to the client.

If there is not enough free disk space to store the new object (or additional copy when caching with replication is employed) we invoke a replacement policy. We emphasize at this juncture that our focus is on the impact of the *caching policy* on the proxy performance. The studied caching policies (i.e., caching with object replication and caching without object replication) may be combined with any *replacement policy*. For illustration we consider the simple and well–known Least Recently Used (LRU) and Least Frequently Used (LFU) replacement policies. (We note that a wide variety of replacement policies has been proposed, however, LRU continues to be the very popular policy due to its simplicity.) With LRU replacement we first check whether we can remove one copy of the object that was requested least recently without interrupting ongoing streams. We verify whether the ongoing streams (if any) of the least recently requested object can be supported by the remaining copies (if any). If so, we remove one copy of that object. Otherwise, we move on to the next to least recently requested object and start over. This replacement algorithm terminates when we have freed up enough space to cache the new object or we have considered all cached objects. In the latter case the attempt to cache the new object fails and the object is streamed from the origin server directly to the client.

With LFU replacement a request counter is maintained for every object in the cache. When the replacement policy is invoked we consider first the object with the smallest request counter value, then the object with the next to smallest counter value, and so on.

We have conducted a simulation study of these caching strategies. For the simulation study we generate 1000 video objects from the 10 pseudo traces from Table 2 in the following manner. For each of the 1000 video objects we randomly select one of the 10 pseudo traces and a random average rate between 1.5 and 2.5 Mbps. For each video object we furthermore draw a random starting phase into the selected pseudo trace and a random lifetime from an exponential distribution with mean L video frames. In the simulation client requests arrive according to a Poisson pro-

cess. For each client request one of the 1000 video objects is drawn according to the Zipf distribution with parameter ζ. (The request arrival rate is set to $0.95D \cdot \bar{b}_m/L$, where D is the number of disks in the proxy and $\bar{b}_m$ is the average number of streams that a single disk can support subject to $P_{\text{loss}}^{\text{disk}} \leq 10^{-6}$; for simplicity we assume that each disk stores at most one video object.)

Figure 3 shows the hit rate as a function of the number of disks in the proxy for caching without object replication and caching with object replication both with LRU replacement and LFU replacement (ignore the "Explicit tracking" curves for now). The hit rate is defined as the ratio of the number of requests served out of the cache (without contacting the origin server) to the total number of client requests. (The results for the byte hit rate are similar.) The Zipf parameter of the client request distribution is set to $\zeta = 0.75$ or $\zeta = 1.0$ for this experiment. The average length of the video objects is set to L = 5,000 frames (corresponding to 3.5 minutes) or L = 20,000 frames, (corresponding to 14 minutes). We observe from the plots that caching with object replication outperforms caching without replication by a significant margin for $\zeta = 1.0$; for $\zeta = 0.75$ the margin is less pronounced. Interestingly, we see from Figure 3 that the replacement policy (LRU or LFU) has no impact on the proxy performance.

In Figure 4 we plot the hit rate as a function of the average length of the video objects (which we assume is identical to the stream lifetimes, i.e., clients receive the entire object). We consider proxies with $D = 50$ disks and with $D = 100$ disks and Zipf request patterns with $\zeta = 0.75$ and $\zeta = 1.0$ in this experiment. The figure reveals that for short–lived streams (< 2000 video frames, corresponding to roughly 1.3 minutes) LFU replacement outperforms LRU replacement. We also see that the caching policy (caching with or without object replication) has no impact on the proxy performance. As the streams become longer lived, however, the replacement policy looses its impact on the proxy performance and the caching policy becomes dominant (especially for the more localized access pattern, ζ = 1.0, and the large proxy, D = 100). The reason for this is that, roughly speaking, it becomes harder to find an object copy that can be removed without interrupting ongoing streams when the streams become longer lived. As a result both replacement policies tend to pick the same object for removal. The main conclusion from this experiment is that object replication is not needed for caching of text and image objects (which can be thought of as having a lifetime of zero). However, for caching of continuous media objects, replication is crucial for good proxy performance, especially when a large proxy serves a client community with a highly localized access pattern.

We next investigate how well the caching policies adapt to different client request patterns. In Figure 5 we plot the hit rate as a function of the Zipf parameter of the request distribution. In this experiment we consider proxies with $D = 50$ disks and $D = 100$ disks and the average object length is set to L = 5,000 frames or L = 20,000 frames. The plots clearly show that caching with object replication outperforms caching without object replication for Zipf request parameters larger than 0.65.

Note that the four discussed and evaluated caching strategies (caching with and without object replication, both with LRU and LFU replacement) implicitly track the client request pattern. Popular objects — once cached — tend to stay in the cache since it is very likely that their removal would interrupt ongoing streams. In addition, caching with object replication is able to adapt to highly localized request patterns as it tends to cache more copies of objects that are relatively more popular. A shortcoming of the implicit tracking schemes is that they do not directly take the objects' popularities into consideration. We observed in our simulations that quite frequently, moderately popular objects fill up the cache and prevent very popular objects from being cached. We are therefore motivated to explicitly take the objects' popularities into consideration.

5.2 Explicit Tracking

Our explicit tracking approach uses an exponential weighted moving average to estimate the client request pattern. Based on the estimated request pattern we determine which objects (and how many copies thereof) to cache in the proxy.

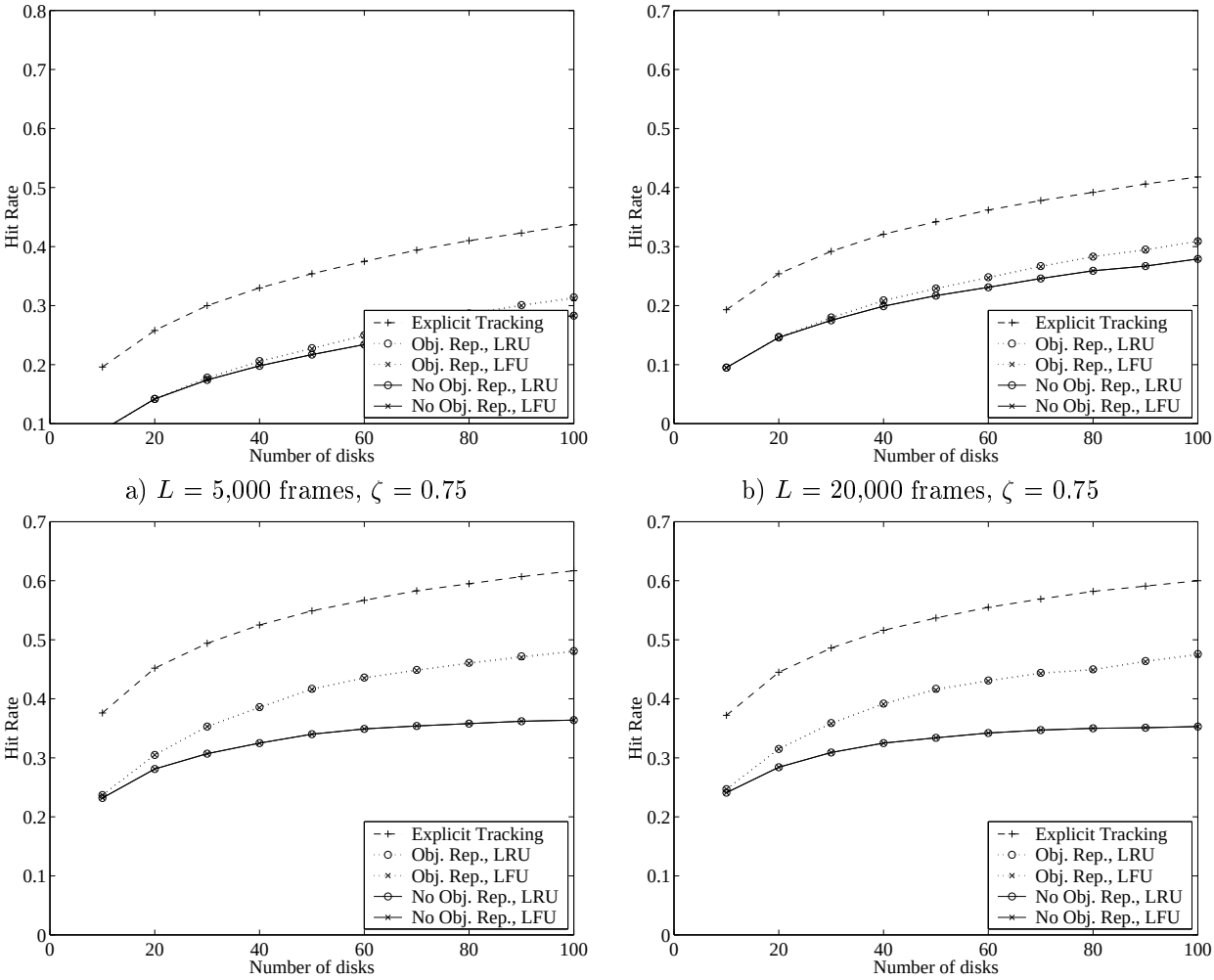
The estimation procedure works as follows. The proxy maintains estimates $\hat{r}_m$ of the request rate (requests per time slot of length Δ, we set $\Delta = 1$ minute in our numerical work) for all objects m, $m = 1, \ldots, M$, requested by its client community. The estimates $\hat{r}_m$ are updated at the end of every time slot. Let req_m denote the number of requests for object m in the just expired time slot. The estimates $\hat{r}_m$ are updated according to

$$\hat{r}_m \leftarrow (1 - \alpha_m)\hat{r}_m + \alpha_m req_m,$$

where α_m is an object specific dampening factor. We set α_m such that $(1 - \alpha_m)^{\tau_m} = 1/e$, where τ_m is the "aging time" (in multiples of the slot length) of object m. We propose to set τ_m to a small value for objects that "age" relatively quickly, such as news clips. On the other hand, τ_m should be set to a large value for objects that "age" slowly, such as on–line lectures, operating instructions or video clips showcasing products in on–line shopping systems. In our numerical work we set τ_m = 10,000 minutes ($\approx$ 7 days) for all objects.

Based on the estimated request rates $\hat{r}_m$, $m = 1, \ldots, M$, we calculate the popularity estimates $\hat{q}_m$ as

$$\hat{q}_m = \frac{\hat{r}_m \cdot M}{\sum_{l=1}^{M} \hat{r}_l}.$$



a) L = 5,000 frames, $\zeta = 0.75$

b) L = 20,000 frames, $\zeta = 0.75$

c) L = 5,000 frames, $\zeta = 1.0$

d) L = 20,000 frames, $\zeta = 1.0$

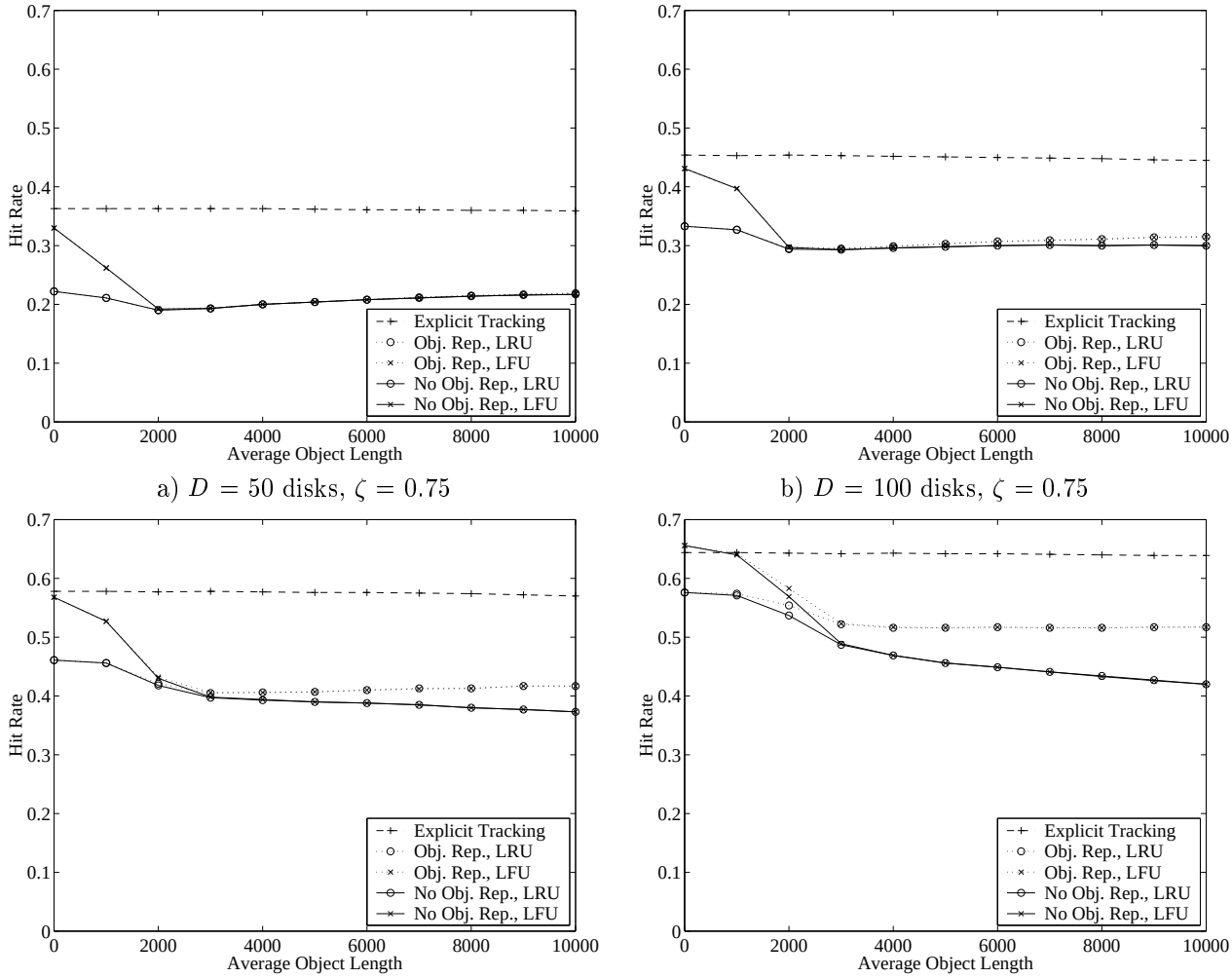
Figure 3: Impact of proxy server resources.

We use the popularity estimates $\hat{q}_m$ to decide which objects (and how many copies thereof) to cache. Our caching policy strives to match the distribution of the number of copies of the cached objects to the estimated popularities. Formally, let $\hat{C}_m, m = 1, \ldots, M$, denote the number of copies of object m required to match the estimated popularities $\hat{q}_m$. Furthermore, let $C_m,\ m = 1, \ldots, M$, denote the number of copies of object m that are currently in the cache. For reasons that will become clear shortly, we distinguish between the number of copies C_m in the cache and the number of copies $\hat{C}_m$ suggested by the popularity estimates. The $\hat{C}_m$'s are matched to the $\hat{q}_m$'s with the following replication algorithm:

1. $\hat{q}_m = \hat{r}_m \cdot M / \sum_{l=1}^{M} \hat{r}_l, \quad m = 1, \ldots, M.$
2. $\hat{C}_m = \lfloor \hat{q}_m D \rfloor, \ m = 1, \ldots, M.$
3. Calculate $\hat{C} = \hat{C}_1 + \cdots + \hat{C}_M.$
4. If $\hat{C} < D$, increment $\hat{C}_m$ for the most popular object, then for the next most popular object, and so on, until $\hat{C} = D$.

Algorithm 2: Replication algorithm.

The storage overhead of this popularity estimation procedure is $O(M)$. This is because the estimate $\hat{r}_m$, the counter req_m, the popularity estimate $\hat{a}_m$, and the object replication have to be maintained for every object $m,\ m = 1, \ldots, M$, requested by the proxy's client com-



a) $D = 50$ disks, $\zeta = 0.75$

b) $D = 100$ disks, $\zeta = 0.75$

c) $D = 50$ disks, $\zeta = 1.0$

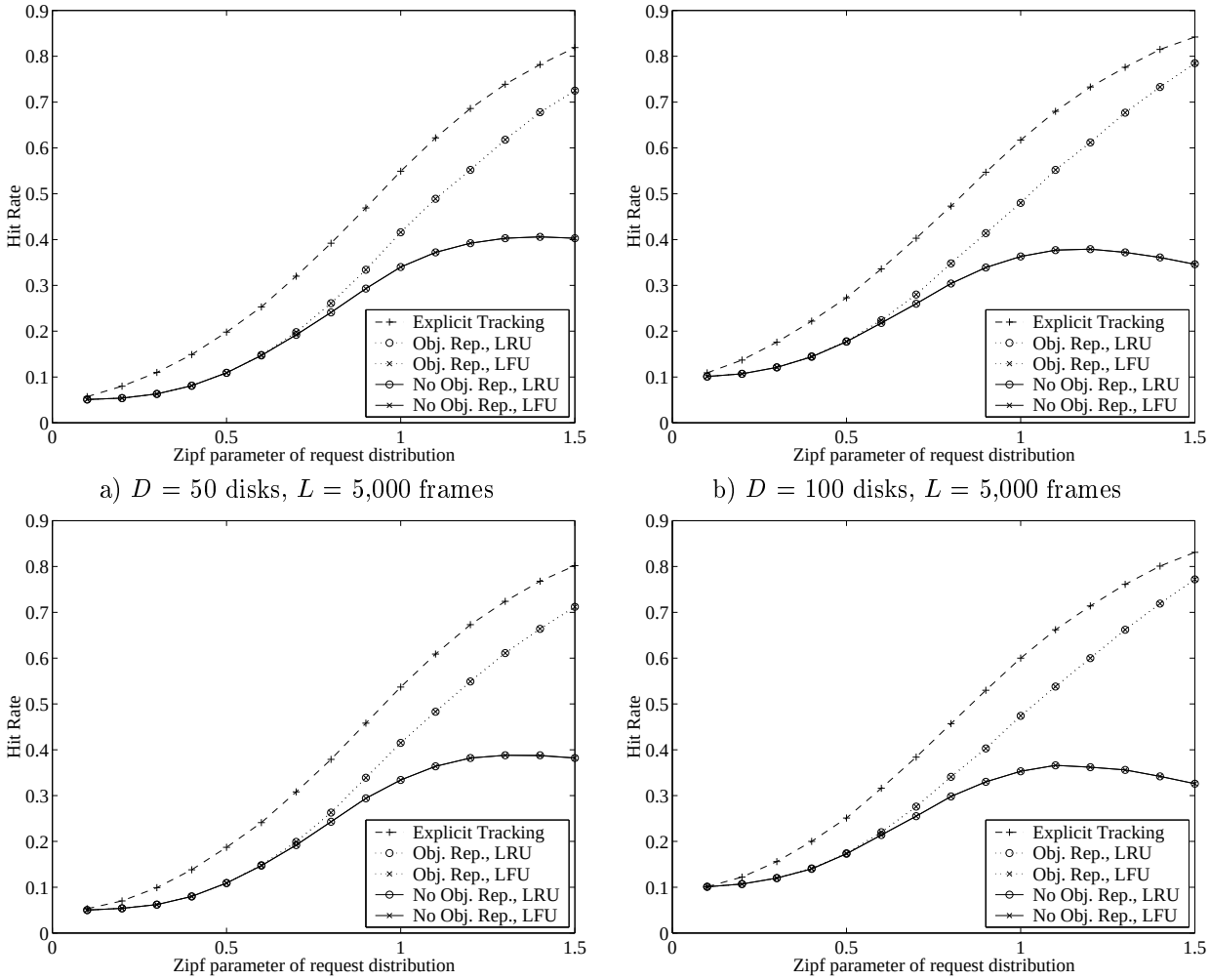
d) $D = 100$ disks, $\zeta = 1.0$

Figure 4: Impact of average object length.

munity. The time complexity of the replication algorithm is approximately $O(M)$. To see this, note that M iterations are required to compute the popularity estimates $\hat{a}_m$ and matching replication $\hat{C}_m$. At most D iterations of Step 4 are required until $\hat{C} = D$ is satisfied, however, typically $D \ll M$. For simplicity we assume in the replication algorithm that each disk stores exactly one video object. Note that this replication algorithm differs from the replication algorithm of Section 4 in that it caches only objects with $\hat{q}_m \geq 1/D$. Based on the $\hat{C}_m$'s obtained with this replication algorithm we propose a caching policy for explicit tracking.

Similar to the implicit strategies discussed in the previous section, the caching policy for explicit tracking is invoked whenever the proxy can not satisfy a client's streaming request. This is the case when either (1) the requested object j^* is not cached (i.e., $C_{j^*} = 0$), or (2) the requested object j^* is cached but the additional stream can not be supported by the cached copies $C_{j^*} \geq 1$ without violating the QoS requirement $P_{\text{loss}}^{\text{disk}} \leq \epsilon$. Our caching policy with explicit tracking works as follows. First, we execute the replication algorithm to determine the current popularity estimates $\hat{q}_m$ and the matching object replication $\hat{C}_m$, $m = 1, \ldots, M$. If $\hat{C}_{j^*} \leq C_{j^*}$ we do not attempt to cache object j^* and it is streamed from the origin server directly to the client. Otherwise, i.e., if $\hat{C}_{j^*} > C_{j^*}$, we attempt to store a copy of object j^* in the disk array. In case (1) this is the first copy of object j^*, which is obtained via the wide area network from the appropriate origin server. In case (2) this is an additional copy of object j^*, which is generated internally from the other already cached copy. If there is enough empty disk space in the proxy we place the new/additional copy of object j^* there, otherwise we invoke the replacement policy.

Roughly speaking, the replacement policy tries to remove one copy of an object that has more copies in the



a) $D = 50$ disks, $L = 5{,}000$ frames

b) $D = 100$ disks, $L = 5{,}000$ frames

c) $D = 50$ disks, $L = 20{,}000$ frames

d) $D = 100$ disks, $L = 20{,}000$ frames

Figure 5: Impact of object request distribution.

cache than are required to match its popularity. Formally, let $\mathcal{R} = \{j : C_j > \hat{C}_j,\ 1 \leq j \leq M,\ j \neq j^*\}$. If $\mathcal{R}$ is non–empty we pick some $j \in \mathcal{R}$ and check whether we can remove one copy of object j without interrupting ongoing streams. This amounts to verifying whether the ongoing object–j streams (if any) can be supported by the remaining $C_j - 1$ copies. If so, we remove one copy of object j and the replacement algorithm terminates if enough disk space has been freed up. Otherwise, we remove object j from consideration by setting $\mathcal{R} \leftarrow \mathcal{R} - \{j\}$ and start over. The replacement algorithm terminates when we have freed up enough space or end up with an empty $\mathcal{R}$. In the latter case the attempt to cache object j^* fails and object j^* is streamed from the origin server directly to the client.

The results of our simulation study of the explicit tracking scheme are given in Figures 3, 4 and 5. The plots show that the explicit tracking scheme consistently outperforms the implicit tracking schemes with LRU or LFU replacement. We observe from Figure 4 that the gap in performance widens as the average object length increases; explicit tracking achieves roughly 18 % higher hit rates than the implicit tracking schemes when the average object length exceeds 2000 video frames. Also, we observe from Figure 5 that explicit tracking outperforms the other schemes for all requests patterns. In summary, we find that explicit tracking is a very attractive caching strategy for continuous media objects.

Finally, we investigate the impact of the proposed caching strategies on the utilization of the disk array. We note, however, that the disk array utilization is only an auxiliary performance metric of a caching strategy; the hit rate is the decisive performance metric for a caching strategy. In our utilization analysis we focus on the utilization of the disk array bandwidth We define the cache utilization as the long run ratio of the sum of the average rates of the streams supported by the proxy to the proxy's total disk bandwidth

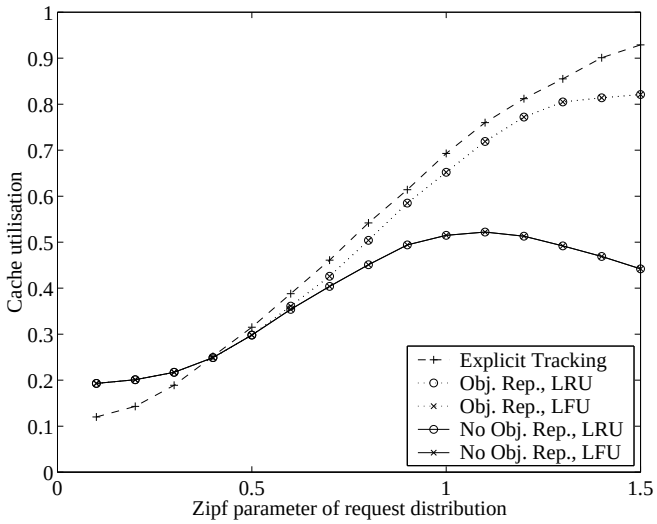


Figure 6: Cache utilization for varying Zipf parameter

(obtained by summing the disk bandwidths given by Eqn. (1)). Figure 6 gives the cache utilization as a function of the Zipf parameter of the request pattern. We consider a proxy with $D = 50$ disks and movies with an average length of $L = 20{,}000$ frames in this experiment. The plots indicate that caching with object replication achieves higher cache utilizations than caching without object replication; explicit tracking gives even higher utilizations. The cache utilization results strengthen our conclusion of the importance of object replication and explicit popularity tracking.

6 Conclusion

We have studied caching strategies for continuous media objects in this paper. The basis for our study is our model for VBR video streaming that provides statistical QoS. We find that for caching of continuous media objects, conventional caching without object replication achieves only small hit rates. We have proposed novel caching strategies that either implicitly or explicitly track the client request pattern. Our numerical evaluation indicates that these novel caching strategies achieve significantly higher hit rates for continuous media objects. In our ongoing research we study refinements of the explicit tracking scheme, such as a refined replacement algorithm, which tries to remove one copy of objects with more than one cached copy before considering objects with only one cached copy.

References

[1] Streaming media caching white paper. Technical Report, Inktomi Corporation, 1999. `http://www.inktomi.com/products/-traffic/streaming.html`.

[2] Y. Birk. Random RAIDs with selective exploitation of redundancy for high performance video servers. In *Proceedings of NOSSDAV '97*, St. Louis, Missouri, May 1997.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf–like distributions: Evidence and implications. In *Proc. of IEEE INFOCOM*, pages 126–134, New York, NY, March 1999.

[4] I. Dalgic and F. A. Tobagi. Characterization of quality and traffic for various video encoding schemes and various encoder control schemes. Technical Report CSL–TR–96–701, Stanford University, Departments of Electrical Engineering and Computer Science, August 1996.

[5] A. Dan, D. Sitaram, and P Shahabuddin. Dynamic batching policies for an on–demand video server. *Multimedia Systems*, 4(3):112–121, June 1996.

[6] W. Feng and J. Rexford. A comparison of bandwidth smoothing techiniques for the transmission of prerecorded compressed video. In *Proceedings of IEEE Infocom*, pages 58–67, Kobe, Japan, April 1997.

[7] J. Gafsi and E. W. Biersack. Data striping and reliability aspects in distributed video servers. In *Cluster Computing: Networks, Software Tools, and Applications*, 1998. Available at `http://www.eurecom.fr/~erbi`.

[8] J. Gafsi and E.W. Biersack. Data striping and reliability aspects in distributed video servers. *Cluster Computing: Networks, Software Tools and Applications*, February 1999.

[9] J. Gafsi and E.W. Biersack. Performance and reliability study for distributed video servers: Mirroring or parity. In *Proc. of IEEE Int. Conf. on Multimedia Computing and Systems*, pages II 628–634, June 1999.

[10] M. W. Garret. *Contributions toward Real-Time Services on Packet Networks*. PhD thesis, Columbia University, May 1993. ftp address and directory of the used video trace: bellcore.com /pub/vbr.video.trace/.

[11] D. J. Gemmel, H. M. Vin, D. D. Kandalur, P. V. Rangan, and L. A. Rowe. Multimedia storage servers: A tutorial. *IEEE MultiMedia*, 28(5):40–49, May 1995.

[12] D.J. Gemmell, H.M. Vin, D.D. Kandlur, P.V. Rangan, and L.A. Rowe. Multimedia storage servers: A tutorial. *IEEE Computer*, 28(5):40–49, May 1995.

[13] G. A. Gibson, J. S. Vitter, and J. Wilkes. Storage and I/O Issues in Large–Scale Computing . In *ACM Workshop on Strategic Directions in Computing Research, ACM Computing Surveys*, 1996. `http://www.medg.lcs.mit.edu/doyle/sdcr`.

[14] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A simple and efficient service for multiple time–scale traffic. In *Proceedings of ACM SIGCOMM*, pages 219–230, August 1995.

[15] E. Knightly and N. Shroff. Admission control for statistical QoS: Theory and practice. *IEEE Network*, 13(2):20–29, 1999.

[16] M. Krunz, R. Sass, and H. Hughes. Statistical characteristics and multiplexing of MPEG streams. In *Proc. of IEEE INFOCOM*, pages 455–462, April 1995.

[17] B. Özden, R. Rastogi, and A. Silberschatz. Disk striping in video server environments. In *Proceedings of IEEE Conference on Multimedia Systems*, pages 580–589, Hiroshima, Japan, June 1996.

[18] Ni, Yang, and Tsang. Source modelling, queueing analysis and bandwidth allocation for VBR MPEG–2 video traffic in ATM networks. *IEE Proceedings on Communications*, 143(4):197–205, August 1996.

[19] B. Ozden, R. Rastogi, and A. Silberschatz. On the design of a low–cost video–on–demand storage system. *Multimedia Systems*, 4(1):40–54, 1996.

[20] A. L. N. Reddy and J. C. Wyllie. I/O issues in a multimedia system. *Computer*, 27(3):69–74, March 1994.

[21] M. Reisslein and K. W. Ross. Call admission for prerecorded sources with packet loss. *IEEE Journal on Selected Areas in Communications*, 15(6):1167–1180, August 1997.

[22] M. Reisslein and K. W. Ross. High–performance prefetching protocols for VBR prerecorded video. *IEEE Network*, 12(6):46–55, Nov/Dec 1998.

[23] M. Reisslein, K.W. Ross, and S. Shrestha. Striping for interactive video: Is it worth it? In *Proc. of IEEE Int. Conf. on Multimedia Computing and Systems*, pages II 635–640, Florence, Italy, June 1999.

[24] R. Rejaie, M. Handley, and D. Estrin. Architectural considerations for playback of quality adaptive video over the internet. In *submitted for review*, November 1998.

[25] R. Rejaie, M. Handley, and D. Estrin. RAP: An end–to–end rate–based congestion control mechanism for realtime streams in the internet. In *Proc. of IEEE INFOCOM*, pages 1337–1345, New York, NY, March 1999.

[26] R. Rejaie, M. Handley, H. Yu, and D. Estrin. Proxy caching mechanism for multimedia playback streams in the internet. In *submitted for review*, January 1999.

[27] J. Roberts, U. Mocci, and J. Virtamo (Eds.). *Broadband Network Traffic: Performance Evaluation and Design of Broadband Multiservice Networks, Final Report of Action COST 242, (Lecture Notes in Computer Science, Vol. 1155)*. Springer Verlag, 1996.

[28] O. Rose. Statistical properties of MPEG video traffic and their impact on traffic modelling in ATM systems. Technical Report 101, University of Wuerzburg, Institute of Computer Science, Wuerzburg, Germany, February 1995.

[29] S. Sahu, P. Shenoy, and D. Towsley. Design considerations for integrated proxy servers. In *Proc. of Int. Workshop on Network and Operating System Support for Digital Audio and Video*, Basking Ridge, NJ, June 1999.

[30] S. Sahu, Z.L. Zhang, J. Kurose, and D. Towsley. On the efficient retrieval of VBR video in a multimedia server. In *Proc. of IEEE Int. Conf. on Multimedia Computing and Systems*, Ottawa, Canada, June 1997.

[31] J. Salehi, Z. Zhang, J. Kurose, and D. Towsley. Supporting stored video: Reducing rate variability and end–to–end resource requirements through optimal smoothing. *IEEE/ACM Transactions on Networking*, 6(4):397–410, August 1998.

[32] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (RTSP). Request for Comments (Proposed Standard) 2326, Internet Engineering Task Force, April 1998.

[33] S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *Proc. of IEEE INFOCOM*, pages 1310 – 1319, New York, NY, March 1999.

[34] P. J. Shenoy and M. Vin. Efficient striping techniques for multimedia file servers. In *Proceedings of NOSSDAV '97*, pages 25–36, May 1997.

[35] P.J. Shenoy and H.M. Vin. Efficient striping techniques for multimedia file servers. In *Proc. of Int. Workshop on Network and Operating System Support for Digital Audio and Video*, pages 25–36, St. Louis, Missouri, May 1997.

[36] Seagate Disk Detailed Specifications. Disk model medalist 13032. http://www.seagate.com/cda/disk/mark/detail/0,1250,152,00.shtml, 1999.

[37] W. S. Tan, N. Duong, and J. Princen. A comparison study of variable bit rate versus fixed bit rate video transmission. In *Australian Broadband Switching and Services Symposium*, pages 134–141, 1991.

[38] R. Tewari, H.M. Vin, A. Dan, and D. Sitaram. Resource–based caching for web servers. In *Proc. of SPIE/ACM Conf. on Multimedia Computing and Networking*, San Jose, 1998.

[39] Y. Wang, Z. Zhang, D. Du, and D. Su. A network–conscious approach to end–to–end video delivery over wide area networks using proxy servers. In *Proc. of IEEE INFOCOM*, pages 660 – 667, San Francisco, CA, April 1998.

[40] G. K. Zipf. *Human Behavior and Principle of Least Effort: An Introduction to Human Ecology*. Addison–Wesley, Cambridge, MA, 1949.

Martin Reisslein is an Assistant Professor in the Department of Electrical Engineering at Arizona State University, Tempe. He is affiliated with ASU's Telecommunications Research Center. He received the Dipl.-Ing. (FH) degree from the Fachhochschule Dieburg, Germany, in 1994, and the M.S.E. degree from the University of Pennsylvania, Philadelphia, in 1996. Both in electrical engineering. He received his Ph.D. in systems engineering from the University of Pennsylvania in 1998. During the academic year 1994-1995 he visited the University of Pennsylvania as a Fulbright scholar. From July 1998 through October 2000 he was a scientist with the German National Research Center for Information Technology (GMD FOKUS), Berlin. While in Berlin he was teaching courses on performance evaluation and computer networking at the Technical University Berlin. He has served on the Technical Program Committees of IEEE Infocom, IEEE Globecom, and the IEEE International Symposium on Computer and Communications. He has organized sessions at the IEEE Computer Communications Workshop (CCW). His research interests are in the areas of Internet Quality of Service, wireless networking, and optical networking. He is particularly interested in traffic management for multimedia services with statistical Quality of Service in the Internet and wireless communication systems.

Keith W. Ross received his Ph.D. from the University of Michigan in 1985 (Program in Computer, Information and Control Engineering). He was a professor at the University of Pennsylvania from 1985 through 1997. At the University of Pennsylvania, his primary appointment was in the Department of Systems Engineering and his secondary appointment was in the Wharton School. He joined the Multimedia Communications Dept. at Institut Eurecom in January 1998, and became department chairman in October 1998. In Fall 1999, while remaining a professor at Institut Eurecom, he co-founded and became CEO of Wimba.

Keith Ross has published over 60 papers and written two books. He has served on editorial boards of five major journals, and has served on the program committees of major networking conferences, including Infocom and Sigcomm. He has supervised more than ten Ph.D. theses. His research and teaching interests include multimedia networking, asynchronous teaching, Web caching, streaming audio and video, and traffic modeling.

Along with Jim Kurose, Keith Ross is the co-author of Computer Networking: A Top-Down Approach Featuring the Internet," Addison-Wesley, 2000.