Interactive Video Streaming with Proxy Servers

Martin Reisslein Felix Hartanto Keith W. Ross
GMD FOKUS Institute Eurecom
{reisslein, hartanto}@fokus.gmd.de ross@eurecom.fr
www.fokus.gmd.de/usr/{reisslein, hartanto} www.eurecom.fr/~ross

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 conventional LRU or LFU based strategies.**

Keywords— **Caching, Continuous Media Object, Replacement Policy, Statistical QoS, Streaming, VBR Video.**

I. 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 [1]. 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 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 replacement policies, such as LRU and LFU, which *implicitly track* the client request pattern. 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 of caching with implicit tracking. Caching with implicit tracking in turn achieves typically 10 % higher hit rates than conventional LRU or LFU replacement.

II. Architecture

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

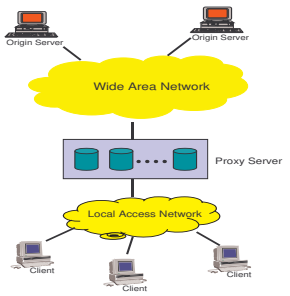

Fig. 1. Architecture for continuous media streaming with proxy server.

origin servers. 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 client directs its request for a particular continuous media object to its assigned proxy server. 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 V) which continuous media object to remove from the cache to make room for the new object. 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.

III. 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. We first develop a *disk model* and derive the effective disk bandwidth for the retrieval of continuous media traffic with tight interactive delay constraints. In [2] we obtain the maximum streaming rate for lossless service:

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

where $retr(I,T)$ denotes the number of bits retrieved for I ongoing streams in one round of length T. The constant l_{seek} is the maximum seek time of the disk and 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. 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. 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

$$P_{\text{loss}}^{\text{disk}} = \frac{E[(X - C_{\text{disk}}T)^+]}{E[X]}, \qquad (2)$$

where X is a random variable which denotes the number of blocks retrieved for I ongoing streams in one round. We evaluate the loss probability with the Large Deviation approximation; see [2] for details.

IV. 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.

Throughout our performance study we assume that the requests for continuous media objects follow the Zipf distribution. For the numerical investigation in this paper we use traces of MPEG encoded video objects. We obtained 10 MPEG–1 traces, which give the number of bits in each encoded video frame, from the public domain. The statistics of the traces are summarized in Table I. 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 I are cached in the proxy's disk array. Furthermore, suppose, that each video object is placed on its own disk. 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. $P_{\text{loss}}^{\text{disk}} \leq 10^{-6}$. The results are reported in Table I (column "St Mux"). 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.

TABLE I
Trace statistics (average rate is 2 Mbps for all traces)

	Frames		Disk Cap.	
Trace	Peak/Mean	Std. Dev.	St. Mux	Peak
bean	13.0	2.25	12	2
bond	10.1	2.03	15	3
lambs	18.3	2.94	11	1
oz	8.4	2.39	14	3
soccer	6.9	1.84	15	4
star wars 1	10.9	2.35	14	2
star wars 2	13.2	2.25	14	2
star wars 3	12.0	2.22	14	2
star wars 4	8.4	2.05	14	3
terminator	7.3	1.79	15	4

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 determine the typical number of streams, that the proxy can simultaneously support; see [2] for details of the evaluation procedure. We find 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 "St. Mux" column of Table I).

The reason for this is that due to the limited disk bandwidth the proxy cannot satisfy many of the requests for the most popular objects. This phenomenon is commonly referred to as *hot–spot problem*. In [2] we study two strategies to overcome the hot–spot problem:

- 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 W ($W \leq D$) disks in the proxy's disk array. This allows the proxy to use the aggregate disk bandwidth of the W disks to stream the video objects. If the video objects are striped over the entire disk array ($W = D$) then the hot–spot problem vanishes and all request distributions can be equally accommodated.

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 I. In the numerical study 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 $\zeta = 1$. 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. We see from the figure that uniform

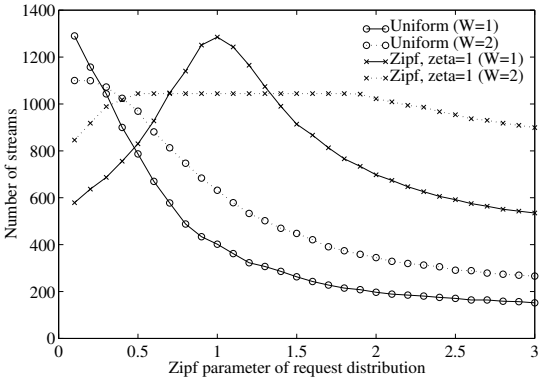

Fig. 2. Robustness of replication strategies.

object replication gives good performance only when the client request pattern is fairly uniform. 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. Because of its simplicity and potential for improved performance we focus on localized placement ($W = 1$) in the next section on caching strategies.

V. 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.

A. 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. In [2] we study two replacement policies, which are based on Least Recently Used (LRU) and Least Frequently Used (LFU) replacement. Figure 3 shows the hit rate as a function of the number of disks in the proxy for LRU replacement and LFU replacement, both with and without object replication (ignore the "Explicit tracking" curves for now). The Zipf parameter of the client request distribution is set to $\zeta = 1$ for this experiment. The average length of the video objects is set to $L =$ 20,000 frames, corresponding to roughly 14 minutes. We observe from the plots that caching with object replication clearly outperforms caching without repli-

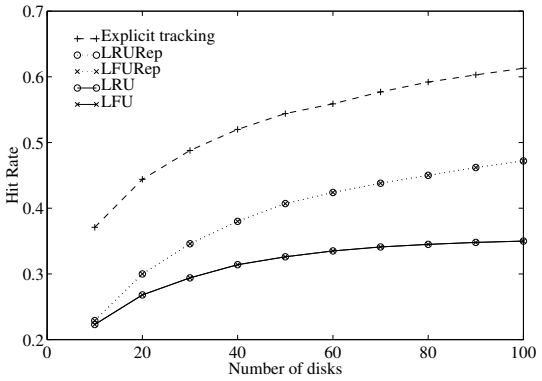

Fig. 3. Impact of proxy server resources.

cation. 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. We consider

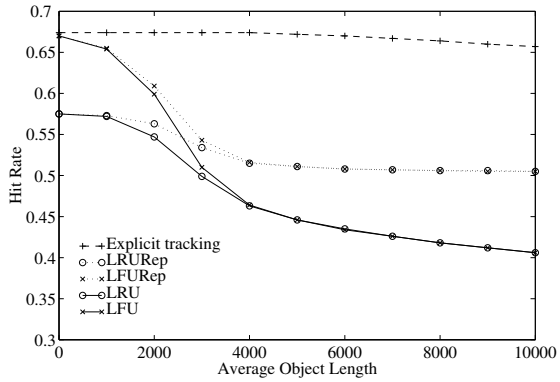

Fig. 4. Impact of average object length.

a proxy with 100 disks and a Zipf request pattern with $\zeta = 1$ 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. 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.

B. 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. Our caching policy strives to match the distribution of the number of copies of the cached objects to the estimated popularities. Let $\hat{C}_m, m = 1, \ldots, M$, denote the number of copies of object m required to match the estimated popularities. Furthermore, let $C_m,\ m = 1, \ldots, M$, denote the number of copies of object m that are currently in the cache.

The caching policy for explicit tracking is invoked 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 a 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 not enough empty disk space in the proxy we invoke a replacement policy, which tries to remove one copy of an object that has more copies in the cache than are required to match its popularity; see [2] for details.

The results of our simulation study of the explicit tracking scheme are given in Figures 3 and 4. The plots show that the explicit tracking scheme consistently outperforms the LRU/LFU based implicit tracking schemes. 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. In summary, we find that explicit tracking is a very attractive caching strategy for continuous media objects.

References

[1] 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`.

[2] M. Reisslein, F. Hartanto, and K. W. Ross, "Interactive video streaming with proxy servers (extended version)," Tech. Rep., GMD FOKUS, Oct. 1999, Available at `http://www.fokus.gmd.de/usr/reisslein` or `http://www.eurecom.fr/~ross`.