Proxy Cache Management for Fine-Grained Scalable Video Streaming

Jiangchuan Liu[1] Xiaowen Chu[2] Jianliang Xu[2]

ljc@cse.cuhk.edu.hk chxw@comp.hkbu.edu.hk xujl@comp.hkbu.edu.hk

[1] Department of Computer Science and Engineering, The Chinese University of Hong Kong

[2] Department of Computer Science, Hong Kong Baptist University

Abstract - **Caching video objects at proxies close to clients has attracted a lot of attention in recent years. To meet diverse client bandwidth conditions, there have been research efforts to combine proxy caching with video layering or transcoding. Nevertheless, these adaptive systems suffer from either coarse adaptation granularity due to the inflexible structures of existing layered coders or high computation overhead due to the transcoding operations. In this paper, we propose a novel adaptive video caching framework that enables low-cost and fine-grained adaptation. The innovative approach employs the MPEG-4 Fine-Grained Scalable (FGS) video with post-encoding rate control. We demonstrate that the proposed framework is both *network aware* and *media adaptive*: clients can be of heterogeneous streaming rates, and the backbone bandwidth consumption can be adaptively controlled. We also examine the design and management issues in the framework, in particular, the optimal video portions for caching and the optimal rate for streaming to each client. Simulation results show that, compared to non-adaptive caching, the proposed framework with optimal cache management not only achieves significant reduction on transmission costs but also enables flexible utility assignment for the heterogeneous clients. Meanwhile, its computational overhead is kept at a low level, implying that it is practically deployable.**

I. Introduction

Due to the increasing demands on video distribution over the Internet, caching video objects at proxies close to clients has attracted much attention in recent years. However, video objects have several distinct features, which make conventional Web caching techniques inefficient, if not entirely inapplicable. In particular, a video object usually has a high data rate and a long playback duration, which combined yield a very high data volume. As an example, a one-hour MPEG-1 video has a data volume of about 675 MB; caching it as a whole is clearly impractical, as several such large streams would exhaust the capacity of a typical cache.

To address these problems, many *partial caching* algorithms have been proposed in recent years [3,4,21], demonstrating that, even if a small portion of a video is stored at the proxy, the network resource requirement can be significantly reduced. Most of these proposals, however, assume that the rate (either constant or variable) of the video is predetermined. Due to this non-adaptability, they suffer from two limitations: first, it is difficult to meet the diverse bandwidth conditions from heterogeneous clients, for a single streaming rate would either overuse or underuse some client bandwidths; second, it is not flexible enough to control the backbone (server-to-proxy) bandwidth consumption, since the streaming rate from the proxy to the clients is not adjustable.

While rate adaptability is a salient feature of video objects, the use of adaptive videos poses great challenges to caching. The problem is particularly complicated by the fact that most conventional rate adaptation mechanisms are executed during the encoding process (*e.g.*, adjusting quantizers [7,18]) and, hence, are difficult to apply to cached videos. There have been research efforts to combine proxy caching with video layering or transcoding [6,13,17]. However, these adaptive systems suffer from either coarse adaptation granularity (due to the inflexible structures of existing layered coders) or high computation overhead (due to the transcoding operations).

In this paper, we propose a novel video caching framework to achieve low-cost and fine-grained rate adaptation. The innovative approach is to employ the MPEG-4 Fine-Grained Scalable (FGS) video with bit-plane coding, which enables post-encoding rate control by partitioning the video stream at any specific rate [8]. These operations can be efficiently implemented at the server or at a proxy with fast response and low cost. The proposed framework is both *network aware* and *media adaptive*: clients can be of heterogeneous access bandwidths, and adaptive FGS videos are used to meet the clients' bandwidth conditions and to control the backbone bandwidth consumption.

We examine the critical design and management issues in the proposed framework. Specifically, there is a two-dimensional space to explore for how to cache an FGS video: the length and the rate of the portion to be cached. We stress that the selection must take into account the interactivities in video playback, *i.e.*, the non-uniform access rates of different portions. Moreover, when a cached video is delivered to a client, different streaming rates can be selected as long as the rate is no higher than the client's available bandwidth. Consequently, the proxy management becomes considerably more complex than that for a non-adaptive video based system. We develop efficient solutions to the above problems. The objective is to offer optimal resource utilization as well as satisfactory and fair services to clients.

The performance of the proposed framework with optimal proxy cache management is extensively examined in various aspects. The results demonstrate its superiority over non-adaptive caching schemes for heterogeneous clients. More importantly, they reveal that our FGS video based adaptive caching system incurs low computation overhead and, hence, is practically deployable.

The rest of this paper is organized as follows. In Section II, we introduce the background as well as related work. Section III describes the FGS video based proxy caching system. The problems of optimal proxy management are formulated in Section IV. We also present efficient solutions to the problems. Their performance is evaluated in Section V. In Section VI, we further consider the practical issues for deploying our system. Finally, Section VII concludes the paper and offers some future directions.

II. Background and Related Work

A. Video Caching for Homogeneous Clients

Numerous cache algorithms for Web proxies have been proposed in the past decade [5]. However, as mentioned in Introduction, several distinct features of video objects like large volume make the conventional Web caching algorithms inapplicable. To this end, many segment or interval caching methods have been proposed, with emphases on cache admission and replacement policies. A comprehensive survey can be found in [21]. Due to the static nature of video contents and the disk bandwidth considerations at proxies, semi-statically caching popular video portions over a relatively long time period, rather than dynamically caching them in response to individual client requests, has also been suggested [3,4,15]. Wang *et al.* [3] proposed a video staging scheme, in which only the part of a stream that exceeds a certain cut-off rate (*i.e.*, the bursts of a VBR stream) is cached at the proxy. Sen *et al.* [4] proposed a *prefix caching* algorithm that caches the initial frames of the video stream. This is particularly attractive in reducing the client start-up latency, due to the predictable sequential nature of video accesses.

Most of these algorithms assumed non-adaptive/non-scalable videos and bandwidth homogeneous clients, though some of them also considered work-ahead smoothing to reduce the backbone bandwidth demand for VBR videos. Our work is motivated by these studies, in particular, the staging and prefix caching algorithms, and complements them by focusing on the issues arising from caching scalable videos with heterogeneous clients.

B. Video Caching for Heterogeneous Clients

To handle the heterogeneity of client bandwidths, a straightforward method is to produce replicated video streams of different rates [18]. Though being used in many commercial streaming products, this method suffers from high replication redundancy. Other recent studies have introduced proxy services with active filtering, which reduces the bandwidth of a video object by transcoding [2,18]. However, they usually incur much higher computation overhead due to transcoding operations.

A more efficient method is to use layered coding (also known as *scalable coding*), which compresses a video into several layers [8]: the most significant layer, called the *base layer*, contains the data representing the most important features of the video, while additional layers, called *enhancement layers*, contain the data that progressively refine the quality of the reconstructed video. Layering has been widely used in live video multicast to heterogeneous and isochronous clients [18]. For proxy-assisted streaming with layered videos, Rejaie *et al.* [6] studied cache replacement and perfecting policies with the objective of alleviating congestion for individual clients. Kangasharju *et al.* [13] simplified the system model by assuming the cached contents are semi-static and only complete layers are cached. They developed effective heuristics to maximize the total revenue based on a stochastic knapsack model.

Our work differs from these previous studies in two aspects: First, we focus on the optimal resource allocation for a set of clients, rather than the performance perceived by an individual client. This is an important design issue from a system point of view. Second, the previous studies employed conventional layered coding, where the number of layers is restricted and the rate of each layer is often fixed. For example, Kangasharju *et al.* [13] focused on two layers only (base layer and one enhancement layer). Our work complements them by employing the fine-grained scalable videos to enhance adaptability.

C. Fine-Grained Scalable (FGS) Video

FGS generalizes the conventional layering (scalable coding) methods through a bitplane coding algorithm, which uses embedded representations for the enhancement layer (also called *FGS layer*) [8]. For illustration, there are 64 (8×8) DCT coefficients for each video block of the enhancement layer; all the most significant bits from the 64 DCT coefficients constitute bitplane 0, all the second most significant bits constitute bitplane 1, and so on and so forth. In the output stream, the bitplanes are placed sequentially to reconstruct the coefficients. A post-encoding filter thus can truncate this embedded stream of the enhancement layer to achieve any specified output rate, with negligible mismatches due to block boundary constraints for the truncation [8,12]. We stress that this rate control method has two advantages: (1) like transcoding, it enables fine-grained rate adaptation, but the computation overhead of the filter is much lower; (2) like layered adaptation, it can be applied to stored videos and enables the proxy to adjust the rate of a cached stream at a low cost. Specifically, for narrowband clients, the proxy can reduce the streaming rate using the filter; for wideband clients, the proxy can fetch some uncached portion (*i.e.*, higher-order bitplanes) from the server and assemble it with the cached portion to generate a higher rate stream. Such operations are difficult to implement using transcoding, not only because it has high computation overhead but also because it changes the stream syntax.

FGS coding has been adopted in the MPEG-4 standard and is undergoing active improvement. We have seen proposals that make efficient use of FGS coding for adaptive video streaming [19]. However, they did not consider proxy caching. To our knowledge, the only work that employed FGS videos in caching is [20]. Their focus however was on developing a general cache management framework, in particular, the replacement policies for mixed-media streaming; the FGS coding was used to for the performance evaluation purpose.

III. System Model of FGS video Based Caching

The video streaming system consists of a server that stores a repository of videos, and a set of proxies on the edge of the network. Selected videos are partially cached at the proxies. A client request is first forwarded to a nearby proxy, which intercepts the request and computes a schedule for streaming: the cached portion of the video can be delivered to the client directly; some uncached portion, if needed, will be fetched from the server and then relayed to the client. Although this process is similar to that of many existing systems, our model has three novel features, making it more general and flexible.

First, we consider a relatively more complex network (*e.g.*, Intranet) behind the proxy, instead of a simple LAN assumed in most existing studies. Examples include an enterprise network or a campus network, which remains highly heterogeneous in terms of client access bandwidths, due to such factors as hardware configurations, connection methods (*e.g.*, Ethernet, ADSL, or wireless LAN), and administrative policies (*e.g.*, in a campus network, the access bandwidth provisioned for a faculty member would be higher than that for a student). To reflect such heterogeneity, we assume that there are M classes of clients, and the maximum access bandwidth for a client (or simply *client bandwidth*) of class i is given by c_i , $i = 1,2,...,M$, which is an upper bound on the video streaming rate from the proxy to a client of class *i*. Without loss of generality, we number the classes in ascending order of the client bandwidth, that is, $c_1 \le c_2 \le \cdots \le c_M$.

Second, we assume that the clients could terminate a video playback prematurely after they requested the playback from the beginning of a video. Existing studies on video server workloads [10] have revealed that such *early terminations* occur quite often and, hence, should be considered in system dimensioning. One approach for modeling early termination is to partition a video into two parts: a *prefix* and a *suffix*, where the prefix could be a preview of the video. If a client feels uninterested after watching the prefix, it will terminate the connection; otherwise, it will continue playback by retrieving the suffix. We denote the lengths of the prefix and the suffix by L_t and L_s, respectively. Assume the probability of early terminations is p_{ET} , $0 < p_{ET} < 1$. The probability of a client accessing the entire video (also the probability of accessing the suffix) is thus $1 - p_{ET}$. While here we consider this simple model only, the method can be extended to a multiple-segment case with non-uniform access rates due to such interactivities as VCR-like operations in playback (some preliminary results can be found in [22]).

Finally and most importantly, we advocate scalable adaptive videos in this system, in particular, the MPEG-4 FGS videos. To model an FGS video, we assume that the base layer of the video has a constant rate r_{base} , which cannot be further partitioned along the rate axis. As such, the base layer represents an ensured lowest playback quality. On the other hand, neglecting the effect of block boundaries for bitplane truncation, the enhancement layer can be adaptively partitioned into any given rate using a filter, either at the server or at the proxy. Thus, as illustrated in Fig. 1, a proxy manager can set the streaming rate to a client of class i in the range of $[r_{base}, c_i]$.

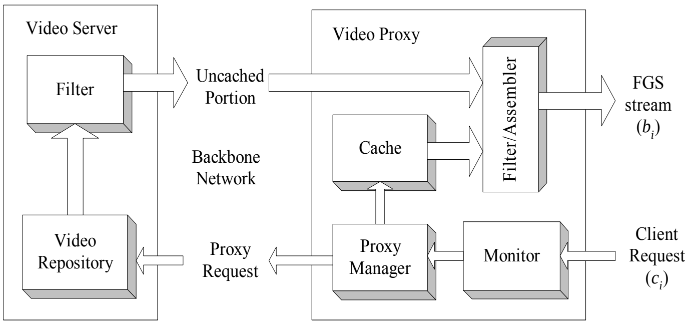


Fig. 1. Functionalities of the video server and a proxy.

The proxy manager also determines which portion of a video is to be cached. In the conventional non-adaptive video caching system, given a cache size for the video, the portion to be cached is simply determined by the length (in terms of playback time). However, with FGS videos, there is one additional dimension to explore: the rate of the portion to be cached. Such flexibility potentially enables better resource utilization, but also complicates the proxy cache management. In addition, there are two cases in which some uncached portion is to be fetched from the server: (1) the length of the demanded portion is longer than that of the cached portion; and (2) the streaming rate is higher than that of the cached portion. In the second case, the uncached bitplanes will be fetched from the server and then assembled with the cached portion to form a stream of higher rate.

As in many previous studies [3,4,14], we assume that the contents of the proxy cache are semi-static and updated periodically with changes of the system workloads. For each period, the key issues in proxy management are to find the optimal length as well as the optimal rate for caching a video, and to determine the streaming rate for each client of the video. To ensure fairness, we let the video streaming rate to any client of class i be identical, denoted by b_i , $b_i \le c_i$, $i = 1,2,...,M$. For a multi-video case of N videos, the proxy manager also needs to determine an optimal resource (cache space and backbone bandwidth) sharing among the videos.

The key parameters of this model are summarized in Table 1. We assume all these parameters are known *a priori* and do not change drastically in a period. Moreover, for ease of exposition, we focus on the interactions among the origin server, a single proxy, and the clients of the proxy. Our results, however, are applicable to the multi-proxy case where each proxy serves a non-overlapping set of clients.

L	Length of the video, $L = L_t + L_s$
L_t	Length of the prefix
L_s	Length of the suffix
p_{ET}	Probability of early terminations
H	Proxy cache size for the video
r_{base}	Base layer rate of the video
λ	Client request rate for the video
M	Number of client classes
p_i	Probability that a client is in class i, $p_i > 0$
c_i	Client bandwidth of class i
b_i	Streaming rate for a class i client ($r_{base} \le b_i \le c_i$)
α_i	Utility of a client in class i, $\alpha_i = b_i / c_i$
$\hat{V}$	Volume of the video with rate c_M, $\hat{V} = L \cdot c_M$
$\hat{B}$	Backbone bandwidth consumption with ideal client utility assignment and no caching

Table 1. Model parameters for a single video. For the multi-video case of N videos, a superscript (k) is to be added to each parameter of video k, $k = 1,2,...,N$.

IV. The Proxy Cache Management Problem and Solutions

In this section, we consider the proxy cache management problem for a single video object as well as for multiple heterogeneous videos. Our objective in designing the proxy cache management module is two-fold: first, to maximize the client utility, which is related to the streaming rate to each class of clients; and second, to minimize the transmission cost, as the video streaming imposes very high data delivery demands to the network. As suggested in previous studies [3], we assume that the local (*i.e.*, proxy-to-client) transmission cost is trivial, and the overall transmission cost is a non-decreasing function of the average backbone (*i.e.*, server-to-proxy) bandwidth consumption. Note that the above two objectives could conflict, because it is obvious that the higher the streaming rate (which results in a higher client utility), the more the uncached portion to be fetched from the origin server (which incurs a higher transmission cost). As such, it is important to find a trade-off between them.

A. Minimizing Transmission Cost

We first consider the transmission cost for a single video with specified streaming rates for each class of clients, b_1, b_2, ..., b_M, $b_i \le c_i, i = 1,2,...,M$. We attempt to answer the following question: Given a limited cache size H to the video, which portion of the stream is cached (referred to as a *caching scheme*) such that the transmission cost is minimized? As said, this objective is equivalent to minimizing the backbone bandwidth consumption.

Note that the rates for different positions of the cached portion can be different when using FGS videos. Denote the rate for position l (measured in the elapsed time from the beginning of the video) of the cached video by $r(l), l \in [0, L]$. A caching scheme for the video is therefore uniquely determined by the shape of $r(l)$. We say that the scheme is *valid* if 1) $r(l)dl \le H \int_0^L$ and 2) for any $l \in [0...L]$, $r_{base} \le r(l) \le c_M$ or $r(l) = 0$. The above two constraints follow the cache size limit and the base layer rate limit, respectively.

A caching scheme is optimal if it is valid and yields the minimum backbone bandwidth consumption for fetching the uncached portion. Let $\hat{V}$ denote the total volume of the video with rate c_M (the maximum client bandwidth). Obviously, if $H \ge \hat{V}$, the caching scheme $r(l) = c_M$, $l \in [0, L]$ is optimal. For the case of limited cache size, we show two lemmas that facilitate searching the optimal caching scheme.

Lemma 1: If $H \le r_{base} \cdot L_t$, the caching scheme $r(l) = \begin{cases} r_{base}, & l \in [0, H / r_{base}] \\ 0, & l \in (H / r_{base}, L] \end{cases}$ is optimal.

Proof: This scheme is clearly valid. For any client request, a video portion of volume H is saved from being transmitted over the backbone. This is the maximum saving per client request that a valid caching scheme could achieve under cache size H. □

Lemma 2: If $r_{base} \cdot L_t < H < \hat{V}$, assuming that the size of the cached prefix is fixed to H_t ($\in [r_{base} \cdot L_t, H]$), there exists an optimal caching scheme:

$$r(l) = \begin{cases} r_t, & l \in [0, L_t] \\ r_s, & l \in (L_t, L_c], \\ 0, & l \in (L_c, L] \end{cases}$$

where r_t and r_s respectively represent (constant) rates of the cached prefix and cached suffix, and are given by $r_t = H_t / L_t$ and $r_s = \max\{r_{base}, (H - H_t) / L_s\}$; L_c is the total length of the cached portion, $L_c = L_t + (H - H_t) / r_s$.

The proof of this lemma can be found in [22]. Let's concentrate on this case ($r_{base} \cdot L_t < H < \hat{V}$) with a given H_t. It is easy to show that $r_t \ge r_s$ for an optimal caching scheme because a cached prefix will serve both early terminated requests and all other requests. The backbone bandwidth consumption for all requests from class i is therefore given by

$$B^{i}_{H,H_t}(b_i) = \begin{cases} \lambda p_i[(1-p_{ET})(b_i L - H) + p_{ET}(b_i L_t - H_t)], & \\ & r_t \le b_i \le c_M \quad (1) \\ \lambda p_i(1-p_{ET})[b_i L_s - r_s(L_c - L_t)], & r_s < b_i < r_t \\ \lambda p_i(1-p_{ET}) b_i (L - L_c), & r_{base} \le b_i \le r_s \end{cases}$$

which can be easily validated from Fig. 2 (for $r_s < b_i < r_t$). It follows that the optimal caching scheme for $r_{base} \cdot L_t < H < \hat{V}$ can be accomplished by a one-dimensional search on H_t in the range of $[r_{base} \cdot L_t, H]$. The optimal solution, or the minimum backbone bandwidth consumption, is thus

$$B_H = \min_{r_{base} \cdot L_t \le H_t \le H,\ r_t \ge r_s} \sum\nolimits_{i=1}^{M} B^{i}_{H,H_t}(b_i). \qquad (2)$$

Assume the minimum cache grain is v, which could be the size of a disk block or the size of a GOP (Group-of-Pictures) of the video; the cache size H, as well as H_t, is always a multiple of the grain v. Then the complexity to search B_H is $O(H/v)$.

For $H \ge \hat{V}$ and $H \le r_{base} \cdot L_t$, the optimal caching scheme can also be uniquely represented using r_t, r_s, and L_c. The corresponding backbone bandwidth consumption thus can be calculated by Eqs. 1 and 2 as well.

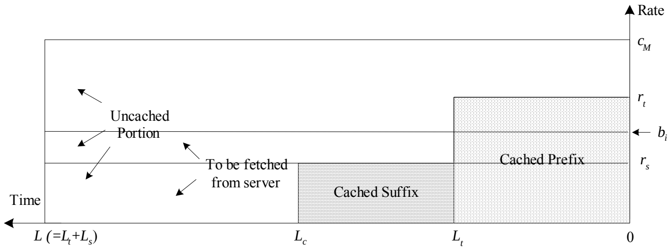


Fig. 2. Illustration of different portions of an FGS video stream.

B. Trading off Backbone Bandwidth with Client Utility

In the above optimization, we assume that the streaming rates for the clients of the video are specified. We now consider a more general and flexible scheme that makes effective use of FGS.

Define the utility of a class *i* client as $\alpha_i = b_i / c_i$ for $r_{base} \le b_i \le c_i$. The ideal utility assignment is $\alpha_i = 1$, $i = 1,2,...,M$, *i.e.*, the client bandwidth is fully utilized for every class. However, it is clear that the higher the client utility, the more the transmission cost or the backbone bandwidth consumption. To reduce bandwidth consumption, one solution is to block some of the client requests. Although this has been suggested in previous studies, it is unfair to the blocked clients. The FGS video, however, offers an alternative method by trading off bandwidth with client utility, that is, assigning a somewhat lower yet acceptable streaming rate to each class of clients.

More explicitly, we would like to investigate whether there exists some utility assignment to each class of clients of the video, such that the backbone bandwidth consumption is no more than $\eta\hat{B}$, where $\hat{B}$ is the backbone bandwidth consumption with the ideal utility assignment and zero-size cache (*i.e.*, no caching); η thus reflects the factor of backbone bandwidth reduction. We are particularly interested in a utility assignment that achieves the best "social welfare", *i.e.*, the total utility of the clients is maximized. This utility optimization problem for the single video can be formally described as follows:

$$\textbf{\textit{MU-SV}}: \quad max \quad U_{H,\eta} = \sum\nolimits_{i=1}^{M} \lambda p_i \alpha_i, \qquad (3)$$
$$s.t. \quad r_{base} / c_i \le \alpha_i \le 1,\ i = 1,2,...,M,$$
$$\alpha_{i-1} c_{i-1} \le \alpha_i c_i, \quad i = 2,3,...,M,$$
$$B_H \le \eta\hat{B},$$

where $U_{H,\eta}$ is the total client utility per unit time for a cache size H and a backbone bandwidth reduction factor η. The constraint $\alpha_{i-1}c_{i-1} \le \alpha_i c_i$ (equivalent to $b_{i-1} \le b_i$) is to preserve the order (priority) of the client classes in resource sharing.

Assume that there is a minimum bandwidth grain w; the backbone bandwidth consumption for a certain class of clients is rounded to a multiple of w. Fig. 3 presents an algorithm to solve the problem instance of a given size of the cached prefix, H_t. ***MU-SV*** thus can be accomplished by a search on H_t, similar to that in the previous subsection.

In this algorithm, $u_{i,j,k}$ represents the maximum total utility per unit time for classes 1 through i, with backbone bandwidth j consumed by classes 1 through i, and backbone bandwidth k consumed by class i. Function $(B^{i}_{H,H_t})^{-1}(k)$ is a generalized inverse of $B^{i}_{H,H_t}(b_i)$ (see Eq. 1), representing the highest streaming rate for a class i client given that the backbone bandwidth consumed by this class is k. A special case is k =0. Recall that we assume $\lambda p_i > 0$ and $1 - p_{ET} > 0$ in the system; zero backbone bandwidth consumption implies that $L_c = L$ and the streaming rate is no higher than r_s. Therefore, if $L_c = L$, we let $(B^{i}_{H,H_t})^{-1}(0)$ be r_s to maximize the total utility; otherwise, we set $(B^{i}_{H,H_t})^{-1}(0)$ to 0 and $B^{i-1}_{H,H_t}(0)$ to $-\infty$. Finally, note that $B^{i}_{H,H_t}(b_i)$ is undefined for $b_i < r_{base}$ and $b_i > c_M$; if directly inversing Eq. 1 yields a value in these two ranges, we set $(B^{i}_{H,H_t})^{-1}(k)$ to 0 and c_M, respectively.

The correctness of this algorithm can be found in [22]. Given the cache grain v, applying the algorithm for each H_t solves problem ***MU-SV*** in time $O(M \cdot \lfloor \eta\hat{B} / w \rfloor^3 \cdot H / v)$. Since the accuracy of this algorithm depends on v and w, we shall investigate their impact in the next section through simulation experiments.

```
$u_{1,j,k} \leftarrow -\infty$  /* Initialization */
for $k = 0$ to $\eta\hat{B}$ step $w$ do
    for $j = k$ to $\eta\hat{B}$ step $w$ do
        $u_{1,j,k} \leftarrow \lambda p_1[(B^1_{H,H_t})^{-1}(k)] / c_1$

for $i = 2$ to $M$ do  /* Table filling */
    for $k = 0$ to $\eta\hat{B}$ step $w$ do
        $y \leftarrow (B^i_{H,H_t})^{-1}(k)$
        for $j = k$ to $\eta\hat{B}$ step $w$ do
            $max \leftarrow -\infty$
            for $x = 0$ to $B^{i-1}_{H,H_t}(y)$ step $w$ do
                $temp \leftarrow \lambda p_i y / c_i + u_{i-1,j-k,x}$
                if $max < temp$
                    then $max \leftarrow temp$ , $bw \leftarrow x$
        $u_{i,j,k} \leftarrow max$ , $h_{i,j,k} \leftarrow bw$

$u^* \leftarrow -\infty$  /* Extract optimal results */
for $bw = 0$ to $\eta\hat{B}$ step $w$ do
    $temp \leftarrow u_{M,\lfloor \eta\hat{B} / w \rfloor \cdot w, bw}$
    if $u^* < temp$
        then $u^* \leftarrow temp$ , $bw^* \leftarrow bw$

$b' = \lfloor \eta\hat{B} / w \rfloor$ , $bw' = bw^*$
for $i = M$ down to $1$ do
    $\alpha_i^* \leftarrow (B^i_{H,H_t})^{-1}(bw') / c_i$
    $b' \leftarrow b' - bw'$ , $bw' \leftarrow h_{i,b',bw'}$

Output:
    $u^*$ : maximum expect utility for the given $H$ , $H_t$ , $\eta$
    $\alpha_i^*$ : corresponding utility assignment of a class $i$ client
```

Fig. 3. Algorithm for an instance of ***MU-SV*** with given H_t .

C. Proxy Management for Multiple Heterogeneous Videos

We now extend the above proxy cache management policies to the case of multiple heterogeneous videos. Assume that there are N videos, indexed as 1 through N . Given the total cache size H^T and the total bandwidth reduction factor η^T for all the videos, our objective is to find a cache partitioning ($H^{(k)}, k = 1,2,...,N$) as well as bandwidth partitioning ($\eta^{(k)}$, $k = 1,2,...,N$) for these videos, such that the system utility (total utility of all the clients for the videos) is maximized. This problem can be formally described as follows:

$$\textbf{\textit{MU-MV}}: \quad max \quad \sum_{k=1}^{N} U^{(k)}_{H^{(k)},\eta^{(k)}} , \tag{4}$$

$$s.t. \quad \sum_{k=1}^{N} H^{(k)} \le H^T ,$$

$$\sum_{k=1}^{N} B^{(k)}_{H^{(k)}} \le \eta^T \sum_{k=1}^{N} \hat{B}^{(k)} .$$

This is a 2-dimesional (bandwidth and space) version of the knapsack problem. Given cache grain v and backbone bandwidth grain w , it can be solved by extending the known pseudo-polynomial time algorithm for 1-D knapsack [1:Ch 16], and the time complexity of this solution is bounded by $O\left(N \cdot H^T \cdot H' \cdot v^{-2} \cdot \left\lfloor \eta^T \sum_{k=1}^{N} \hat{B}^{(k)} \cdot B' \cdot w^{-2} \right\rfloor\right)$, where $B' = \min\{\eta^T \sum_{k=1}^{N} \hat{B}^{(k)}, \max_k \hat{B}^{(k)}\}$ and $H' = \min\{H^T, \max_k \hat{V}^{(k)}\}$ [22].

V. Performance Evaluation

In this section, we evaluate the performance of our FGS-based caching system through simulation experiments. We examine the system along two dimensions: 1) the transmission cost, or backbone bandwidth consumption; and 2) the quality of the delivered streams, *i.e.*, the client utility.

A. System Settings

For the single video case, we assume that there are five classes of clients, and the client bandwidths of the classes are exponentially spaced, *i.e.*, c_1=128 Kbps and $c_i = 2c_{i-1}$ for $i = 2,3,4,5$, which cover the bandwidths of a broad spectrum of access technologies. For the client population distribution among the classes, ($p_1, p_1, ..., p_5$), we have evaluated various settings in our experiments. Due to space limitation, in this paper we present the simulation results for three typical distributions:

(1) *Uniform*: $(0.2,\ 0.2,\ 0.2,\ 0.2,\ 0.2)$;

(2) *S-narrow*: $(0.5, 0.2, 0.15, 0.1, 0.05)$;

(3) *S-wide*: $(0.05, 0.1, 0.15, 0.2, 0.5)$.

The latter two are skewed distributions, respectively dominated by narrowband clients and wideband clients. The lengths of the entire video and the prefix are set to 100 minutes and 20 minutes, respectively. The probability of early terminations is set to 0.3. Although a rigorous evaluation of the client termination behavior is beyond the scope of this work, we note that higher probabilities of early terminations have been observed in reality [10]; in this case, more benefits can be expected from the caching paradigm advocated in our system.

We assume that the requests follow a Poisson arrival process with a mean rate of one request per minute. We normalized the backbone bandwidth by $\hat{B}$ (the backbone bandwidth consumption with ideal utility assignment and no caching) and the cache size by $\hat{V}$ (the total volume of the video with rate c_M) for all presented results. Hence, our conclusions are generally applicable when the parameters are proportionally scaled. Unless explicitly specified, the default cache grain and backbone bandwidth grain are set at $1/200\hat{V}$ and $1/200\hat{B}$, respectively.

For the multiple video case of N objects, we use similar client settings as described above for each single video, and assume that the access probabilities of the videos follow a Zipf distribution, which has been suggested by workload measurements of media servers. With this distribution, the access probability of video k is $(1/k)^{\theta}/\sum_{j=1}^{N}(1/j)^{\theta}$, where θ reflects the skewness among the client populations of the videos.

B. Evaluation Results for Single Video

B.1 Backbone Bandwidth Reduction

In the first set of experiments, we investigate the backbone bandwidth consumption. We are interested in examining the backbone bandwidth reductions by employing the optimal caching scheme, compared to the following two baseline schemes:

(1) *MaxLen*: $r(l) = \begin{cases} r', l \in [0, H/r'] \\ 0, l \in (H/r', L] \end{cases}$, $r' = \max\{r_{base,} H/L\}$;

(2) *MaxRate*: $r(l) = \begin{cases} c_M, l \in [0, H/c_M] \\ 0, \quad l \in (H/c_M, L] \end{cases}$.

These two schemes are non-adaptive because they are not aware of the client bandwidth distributions. They also resemble the caching schemes for a coarse-grained layering of 2 layers, *i.e.*, caching the base layer only and caching both the base and enhancement layers.

Fig. 4 shows the backbone bandwidth reductions for different cache sizes and class distributions. The client bandwidth is assumed to be fully utilized for all classes, that is, $\alpha_i = 1$, $i = 1, 2, ..., M$. We observe remarkable reductions achieved by our optimal scheme over the two baseline schemes, which is generally over 10% and sometimes over 50%. The reduction depends on the class distributions, *e.g.*, for the S-narrow distribution (Fig. 4b), the reduction is particularly high when compared with the MaxRate scheme, as most of the clients have a relatively low bandwidth and hence, caching the stream of the highest rate becomes wasteful. In this case, increasing the length of the cached stream becomes a better alternative. However, compared to our optimal scheme, the MaxLen scheme still suffers from more than 10% bandwidth excess, because it is not flexible in setting the rates for the cached prefix and suffix to better accommodate early terminated requests. On the contrary, with the S-wide distribution (Fig. 4c), since most clients have high bandwidth demands, the MaxRate scheme is better than the MaxLen scheme, but is still suboptimal. Finally, with the Uniform distribution (Fig. 4a), both the MaxLen and the MaxRate schemes are far from satisfactory.

It is worth noting that, compared to the two non-optimal schemes, there is virtually no extra cost in employing the optimal scheme in our FGS video based caching framework. Hence, we believe that our optimal scheme is an effective means to improve the system performance.

B.2 Utility Improvement

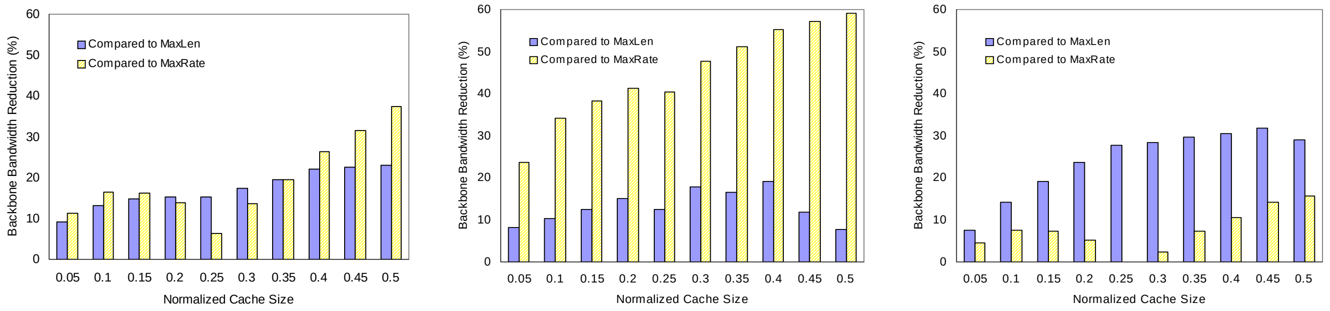
We now examine the tradeoff between the client utility and the backbone bandwidth consumption, given that the streaming rates to the clients can be regulated using the filter/assembler at the proxy. We employ the optimal caching and utility assignment algorithms for our system, as described in Section IV.B. Fig. 5 shows the expected utility for all clients as a function of the backbone bandwidth consumption for different cache sizes and class distributions. Note that the normalized backbone bandwidth is essentially equal to η, as shown in Eq. 3.

It can be seen that, to achieve the optimal utility ($=1$), a relatively high backbone bandwidth is to be consumed if the cache size is very small, *e.g.*, 40% of backbone bandwidth $\hat{B}$ with a cache size $0.3\hat{V}$ for the uniform class distribution (Fig. 4a). However, there is a nonlinear relation between the backbone bandwidth consumption and the client utility. As a result, for the same setting, we can achieve an expected client utility of 0.9 by consuming only 15% of $\hat{B}$, that is, a 10% utility reduction leads to a 62.5% backbone bandwidth reduction. This is particularly evident for the S-narrow distribution, not only because a relatively high volume is cached for the stream to a narrowband client, but also because a slight reduction of the streaming rate to a wideband client will benefit the set of narrowband clients. In this case, the expected utility is over 0.8 even with very limited resources (*e.g.*, backbone bandwidth of $0.05\hat{B}$ and cache size of $0.1\hat{V}$). For the S-wide class distribution, the reduction is not that significant. However, in this case, the absolute backbone bandwidth consumption is much higher than that for the other two distributions; hence, a slight reduction on the normalized bandwidth would still lead to a great reduction in the absolute backbone bandwidth consumed.

Such an adaptive setting of utility clearly offers a flexible space for a designer to explore to either maximize the overall revenue or minimize the overall cost. On the contrary, if the cache size is less than $0.1\hat{V}$ and the backbone bandwidth is less than $0.5\hat{B}$, a non-adaptive system that fixes the client utility to one does not even work for any class distribution.

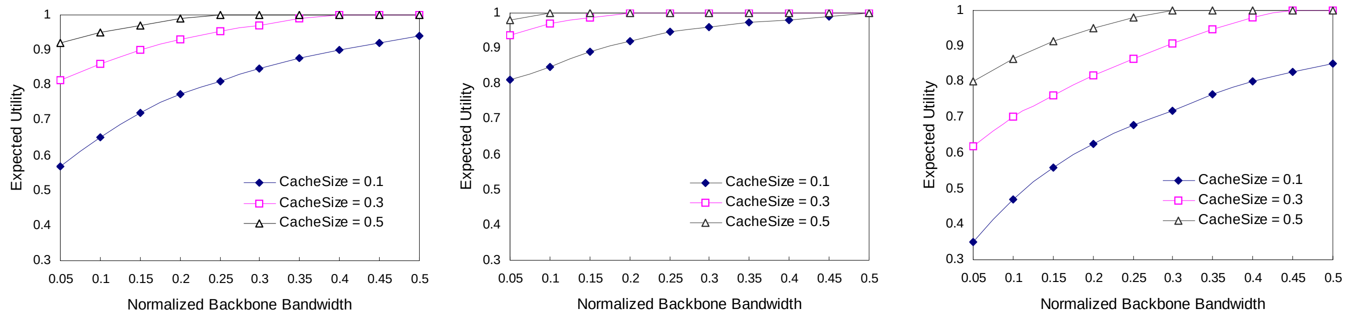
B.3 Sensitivity to Allocation Grains

In our experiments, the default cache grain v and backbone bandwidth grain w are set at $1/200\hat{V}$ and $1/200\hat{B}$, respectively. To investigate their impact, we repeat the experiments of the previous subsection with various grain settings. Fig. 6a shows the maximum gaps in terms of optimal client utilities between each setting and a finer-grained setting: $1/1000\hat{B}$ (bandwidth grain) and $1/1000\hat{V}$ (cache grain). We can see that the performance gap quickly decreases with the refinement of the grains, and the gap for the default setting (1/200 $\hat{V}$ and 1/200 $\hat{B}$) is already lower than 0.008.



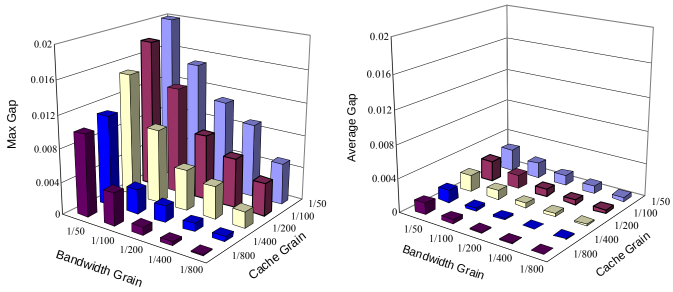
(a) (b) (c)

Fig. 4. Backbone bandwidth reductions achieved by the optimal caching scheme.
(a) Uniform class distribution; (b) S-narrow class distribution; (c) S-wide class distribution.



(a) (b) (c)

Fig. 5. Expected client utility as a function of backbone bandwidth consumption.
(a) Uniform class distribution; (b) S-narrow class distribution; (c) S-wide class distribution.



(a) (b)

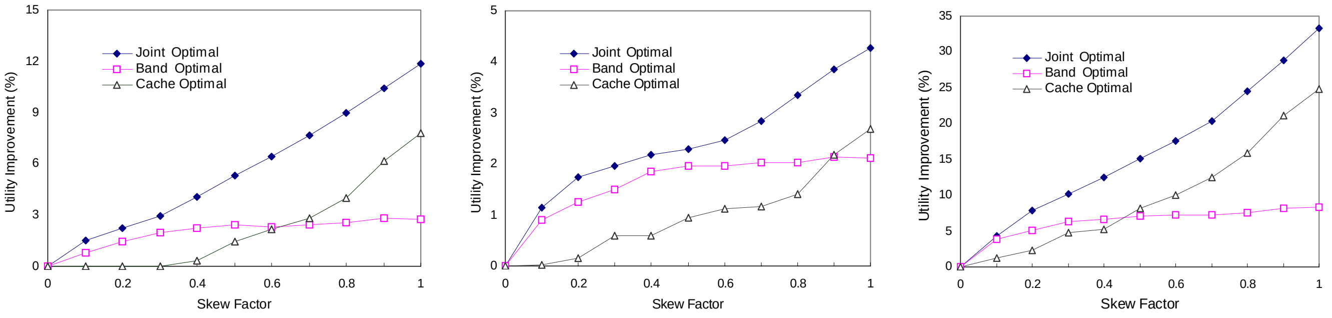
Fig. 6. Performance gaps as compared to a fine-grained setting $v = 1/1000\hat{V}$, $w = 1/1000\hat{B}$. (a) Max gap; (b) Average gap.

This is negligible from a client perception point of view. We also show the average gaps in Fig. 6b, which are in fact much smaller than the corresponding maximum gaps shown in Fig. 6a. Therefore, we believe that the default setting is reasonably good, especially considering that the computation times with this setting are generally less than 30 ms on a common PC (Pentium III 1 GHz).

C. Evaluation Results for Multiple Videos

Finally, we investigate the performance of the proxy management algorithm for multiple videos. We implement the joint optimization scheme ***MU-MV*** for cache and bandwidth partitioning (see Section IV.C), as well as a baseline scheme with a uniform cache partitioning ($H^{(k)} = H^T / N$) among the videos and a bandwidth partitioning proportional to the client population of each video ($\eta^{(k)} = (\lambda^{(k)} / \sum_{j=1}^{N} \lambda^{(j)}) \cdot (\eta^T \sum_{j=1}^{N} \hat{B}^{(i)}) / \hat{B}^{(k)}$). The utility assignment for each single video is optimized using the algorithm for ***MU-SV***. It is easy to verify that the system utility for the baseline scheme is independent of the skew factor θ . Fig. 7 shows the system utility improvement by the joint optimization scheme for $H^T = 0.2\sum_{k=1}^{N} \hat{V}^{(k)}$, and $\eta^T = 0.1$. It is clear that the joint optimization significantly improves the system utility, in particular, with large skew factors, *i.e.*, the client populations of the videos are highly heterogeneous. For the S-wide distribution (Fig. 7c), which has a relatively low system utility for the baseline scheme under the given settings, the utility improvement is over 30% for high skew factors.

To identify respective contributions of the two optimization dimensions (*i.e.*, space and bandwidth), we also show in Fig. 7 the improvements achieved by employing the optimal cache partitioning alone (Cache Optimal) and the optimal bandwidth partitioning alone (Band Optimal). Their corresponding bandwidth and cache partitionings are proportional and uniform, respectively. It can be seen that, though using either optimal partitioning alone also improves the system utility, there remain remarkable gaps as compared to that of the joint optimization, especially for higher skew factors. This is because the utility of each video is a



(a) (b) (c)

Fig. 7. System utility improvement as a function of skew factor θ ($H^T = 0.2\sum_{k=1}^{N} \hat{V}^{(k)}$, $\eta^T = 0.1$) for multi-video allocation. The system utilities of the baseline scheme for the Uniform, S-narrow, and S-wide distributions are 0.77, 0.92, and 0.56, respectively. (a) Uniform distribution; (b) S-narrow distribution; (c) S-wide distribution.

nonlinear function of cache space or backbone bandwidth consumed by the video and, hence, the uniform cache partitioning or the proportional bandwidth partitioning alone is suboptimal for these heterogeneous videos. As such, we believe that it is important to use the joint optimization for the videos.

VI. Comparisons and Practical Considerations

A. Scalable Video or Replicated Video?

As mentioned in Section II, yet another approach to handle the client heterogeneity is stream replication [9,18]; that is, on the server's side, replicated streams are generated to meet the bandwidth condition of each class of clients. Stream replication has several advantages, such as simplicity and compatibility with conventional non-scalable video coders. Hence, it has been widely used in existing commercial streaming systems, *e.g.*, the RealNetwork's SureStream. Nevertheless, replication also leads to data redundancy, not only in server storage but also in proxy cache and bandwidth consumption.

Scalable video and replicated video based streaming systems have been compared in [9] with no proxy caching. A cache-aware comparison was shown in [16]. They considered coarse-grained layering (two layers) and the metric of interest is the request blocking probability. In this set of experiments, we try to complement these works from a client utility and bandwidth consumption point of view with the use of FGS videos.

We first investigate the backbone bandwidth consumptions of the two approaches when the client bandwidth is to be fully utilized. To make a fair comparison, we develop the optimal caching scheme for the stream replication based system. In this system, the cached portion of a stream serves the clients of its own class only, and its rate is equal to the client bandwidth of the class. The problem is thus how to partition a cache with given size *H* for the *M* replicated streams of the video. This is a variation of the cache allocation problem for multiple heterogeneous videos, as solved in [14].

Using the optimal caching schemes, we calculate the backbone bandwidth reduction of the FGS video based system against the stream replication based system, as shown in Fig. 8. For the Uniform and S-narrow distributions, the reduction is significant, often over 40% and sometimes over 60%. For the S-wide distribution, the reduction is relatively smaller because the backbone bandwidth consumption is dominated by the requests from the wideband clients. Nevertheless, the absolute value of saved bandwidth remains high enough in this case, as discussed in Section V-B.2.

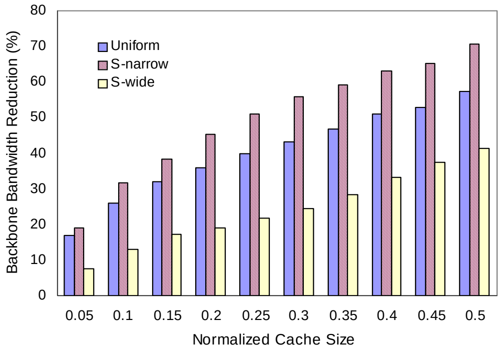


Fig. 8. Backbone bandwidth reduction of FGS video based caching against stream replication based caching for different cache sizes and class distributions.

Next, we consider the case of flexible utility assignment, where the client utility can be lower than one. The problem of optimal caching and client utility assignment for the stream replication based system can be formulated as follows:

$$\textbf{\textit{MU-REP}}: \quad max \quad U_{H,\eta} = \sum_{i=1}^{M} \lambda p_i \alpha_i \ , \qquad (7)$$

$$s.t. \quad r_{\min} / c_i \le \alpha_i \le 1 , \ i = 1,2,\ldots,M ,$$

$$\alpha_{i-1} c_{i-1} \le \alpha_i c_i , \ i = 2,3,\ldots,M ,$$

$$\sum_{i=1}^{M} h_i \le H \text{ , and } B_H \le \eta \hat{B} ,$$

where h_i is the cache size allocated to stream i, and $r_{\min}$ is the lowest streaming rate, which is set to r_{base} in the experiments to ensure a fair comparison. This problem can also be solved by checking different partitionings of the cache and, for each partitioning, the optimal utility assignment can be obtained using an algorithm similar to that for problem ***MU-SV***. Fig. 9 shows the utility improvement of FGS based caching against stream replication based caching for the uniform class distribution. It can be seen that, when the backbone bandwidth consumption is lower than 0.25 $\hat{B}$, the improvement is often higher than 20%. It diminishes with increasing backbone bandwidth, since the client utility becomes saturated (optimal) for both systems. Nonetheless, compared with Figs. 4 and 5, it is clear that the stream replication based system requires far more resources to reach such maximum. For example, with the uniform class distribution, the client utility for our FGS based system has reached one for a cache size of 0.5 $\hat{V}$ and backbone bandwidth consumption of 0.25 $\hat{B}$ (see Fig. 5a); however, according to Figs. 5a and 9, the client utility for replication based system is still below 0.9 in this case.

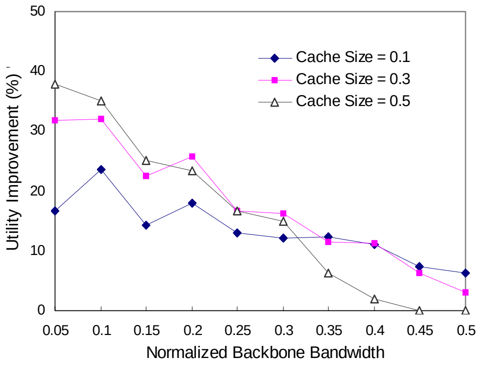


Fig. 9. Utility improvement of FGS video based caching against stream replication based caching for the uniform class distribution.

In the above experiments, we used the function $\alpha_i = b_i / c_i$ to measure client utility. It is worth noting that the FGS coding incurs extra overhead for stream scaling, which potentially leads to quality degradation. A comprehensive study on this issue involves the use of perception-aware utility functions, which is out of the scope of this paper and is indeed an open research problem. However, we are aware that current studies indicate that the quality degradation caused by such overhead is generally less than 10%, where the quality is measured by the Peak Signal-to-Noise Ratio (PSNR) [8]. The degradation can be further minimized using smart reference schemes, such as those introduced in the Progressive FGS [11]. In view of these, and considering that FGS has been adopted by the MPEG-4 standard as well as supported by quite a few industry giants, we believe our FGS video based caching offers a promising cost effective vehicle for video streaming to heterogeneous clients.

B. Efficiency of FGS Filtering and Assembling

Another important practical concern of our system is its computational efficiency. Since the cache allocation and utility assignment are updated periodically and the allocation algorithms are reasonably fast, we mainly focus on examining the efficiency of the filter/assembler module, which is invoked for each client request.

We use a MPEG-4 standard test sequence *News* (CIF) in our experiments. The rate of the sequence is 2 Mbps, and the base layer rate for FGS coding is set to 128 Kbps, which is consistent with our previous experiments. For the sake of comparison, we also examine three other filters that have been widely used in transcoding proxies [2,7,18]: simple frame-dropping (SFD), frame-dropping with drift-compensation (FD-DC), and re-quantization (RQ). A frame-dropping filter discards some B-frames or P-frames to achieve bandwidth reduction. Since a P-frame depends on its preceding I or P frame, to avoid quality drift, if a P frame is discarded, the SFD filter simply discards all subsequent P frames until an I frame arrives. The FD-DC filter avoids this by performing drift compensation for the subsequent P frames. Finally, the RQ filter reduces the stream rate by re-scaling quantizers, which yields better perceptual quality than discarding a whole frame. For each filter/assembler, we repeat filtering/assembling 10 times on the entire sequence, with target rates distributed between 128 Kbps and 2 Mbps.

Fig. 10 shows the average computation times of these various filters as well as the FGS filter/assembler. It can be seen that the computation time of the FGS filter/assembler is about two orders of magnitude lower than that of the FD-DC or RQ filters. Given a frame interval of 30 ms, our PC can support about 300 concurrent filter/assembler modules, and a powerful proxy server would support much more. In this case, the bottleneck of the system is likely the backbone bandwidth, but not the computation power of the proxy. This is however not true for the FD-DC or RQ filters; in particular, the number of concurrent RQ filters cannot be more than 3 on our PC. The SFD filter, though having similar computation time as ours, suffers from the coarse granularity and confined dynamic ranges for rate adaptation as mentioned before.

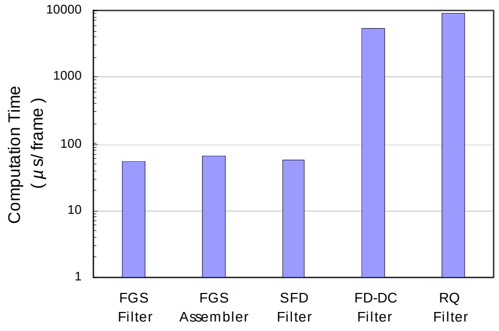


Fig. 10. Average computation time per frame.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we addressed the problem of proxy-assisted video streaming to a set of heterogeneous clients. We proposed an adaptive proxy caching framework using fine-grained scalable (FGS) videos, and explored the benefits associated with FGS in handling client heterogeneity as well as reducing transmission costs. We also developed effective solutions to two important proxy management problems in this framework: which portion to be cached for each FGS video, and which streaming rate to be employed for delivering the stream to each client?

Simulation results showed the proposed framework not only achieves significant backbone bandwidth reduction but also enables flexible utility assignment for heterogeneous clients. Meanwhile, its computation overhead is kept at a low level. We also conducted a systematic comparison between the FGS-based and the replication-based video caching systems, which demonstrated the superiority of the FGS-based caching.

Given the low computation overheads of the FGS filter/assembler, we believe that our rate-adaptive caching framework is practically deployable. However, implementing a whole system remains a challenging undertaking, as it involves not only the optimization of individual components, but also their convergence. There are many possible research issues worth further investigation. For example, what is the impact of the nonlinear relation between video quality and streaming rate? We plan to employ perception-aware utility functions in our future experiments, *e.g.,* those built on the rate-distortion framework. We are also extending our scheme to multiple segments with non-uniform access rates.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Hua Cai at Microsoft Research Asia and Prof. Mihaela van der Schaar at UC Davis for discussions on FGS coding and transcaling. This work was supported in part by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant CUHK C001/2050312 and Grant HKBU FRG/02-03/II-34.

REFERENCES

[1] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[2] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd, “Filters: QoS support mechanisms for multipeer communications,” *IEEE J. Select. Areas Commun.*, vol. 14, no. 7, Sept. 1996.

[3] Y. Wang, Z.-L. 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. IEEE INFOCOM*’98, Apr. 1998.

[4] S. Sen, J. Rexford, and D. Towsley, “Proxy prefix caching for multimedia streams,” in *Proc. IEEE INFOCOM’99*, Mar. 1999.

[5] J. Wang, “A survey of Web caching schemes for the Internet,” *ACM Computer Communication Review*, vol.29, no.5, Oct. 1999.

[6] R. Rejaie, H. Yu, M. Handley, and D. Estrin, “Multimedia proxy caching mechanism for quality adaptive streaming applications in the Internet,” in *Proc. IEEE INFOCOM’00*, Mar. 2000.

[7] D. Wu, Y.T. Hou, and Y.-Q. Zhang, “Transporting real-time video over the Internet: challenges and approaches,” *Proceedings of the IEEE*, vol. 88, no. 12, Dec. 2000.

[8] W. Li, “Overview of the fine granularity scalability in MPEG-4 video standard,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301-317, Mar. 2001.

[9] P. de Cuetos, D. Saparilla, and K. W. Ross, “Adaptive streaming of stored video in a TCP-friendly context: multiple versions or multiple layers,” in *Proc. Packet Video Workshop*, Apr. 2001.

[10] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon, “Analysis of educational media server workloads,” in *Proc. NOSSDAV’01*, June 2001.

[11] F. Wu, S. Li, and Y.-Q. Zhang, “A framework for efficient progressive fine granular scalable video coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, 2001.

[12] H. Radha and M. Schaar, “Partial transcaling for wireless packet video,” in *Proc. Packet Video Workshop*, Apr. 2002.

[13] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross, “Distributing layered encoded video through caches,” *IEEE Trans. Computers*, vol. 51, no. 6, pp. 622-636, June 2002.

[14] B. Wang, S. Sen, M. Adler, and D. Towsley, “Optimal proxy cache allocation for efficient streaming media distribution,” in *Proc. IEEE INFOCOM’02*, June 2002.

[15] S. Jin, A. Bestavros, and A. Iyenger, “Accelerating Internet streaming media delivery using network-aware partial caching,” in *Proc. IEEE ICDCS’02*, July 2002.

[16] F. Hartanto, J. Kangasharju, M. Reisslein, and K. W. Ross, “Caching video objects: layers vs versions?” in *Proc. IEEE Int. Conf. on Multimedia and Expo* (*ICME’02*), Aug. 2002.

[17] X. Tang, F. Zhang, and S. T. Chanson, “Streaming media caching algorithms for transcoding proxies,” in *Proc. 31st Int. Conf. on Parallel Processing* (*ICPP*’02), Aug. 2002.

[18] J. Liu, B. Li, and Y.-Q. Zhang, “Adaptive video multicast over the Internet,” *IEEE Multimedia*, vol. 10, no. 1, Jan. 2003.

[19] T. Kim and M. Ammar, “Optimal quality adaptation for MPEG-4 fine-grained scalable video,” in *Proc. IEEE INFOCOM’03*, Apr. 2003.

[20] F. Yu, Q. Zhang, W. Zhu, and Y.-Q. Zhang, “QoS-adaptive proxy caching for multimedia streaming over the Internet,” *IEEE Trans. Circuits Syst. Video Technol.*, to appear.

[21] J. Liu and J. Xu, “A survey of streaming media caching,” available at: www.comp.hkbu.edu.hk/~xujl/streamCaching.pdf

[22] J. Liu, X. Chu, and J. Xu, “Proxy cache management for fine-grained scalable video streaming,” *Technical Report*, The Chinese University of Hong Kong, Sept. 2003.