CACHING VIDEO OBJECTS: LAYERS VS VERSIONS ?

Felix Hartanto[*] *Jussi Kangasharju*[†] *Martin Reisslein*[‡] *Keith W. Ross*[†]

[*]Dept. of Information Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[†]Institut Eurecom, 2229, route des Cretes, 06904 Sophia Antipolis, France
[‡]Telecommunications Research Center, Dept. of Electrical Engineering, Tempe, AZ 85287–7206

ABSTRACT

Because Internet access rates are highly heterogeneous, many video content providers today make available different versions of the videos, with each version encoded at a different rate. Multiple video versions, however, require more server storage and may also dramatically impact cache performance in a traditional cache or in a CDN server. An alternative to versions is layered encoding, which can also provide multiple quality levels. Layered encoding requires less server storage capacity and may be more suitable for caching; but it typically increases transmission bandwidth due to encoding overhead. In this paper we compare video streaming of multiple versions with that of multiple layers in a caching environment. We examine caching and distribution strategies that use both versions and layers. Our analytical results indicate that mixed distribution/caching strategies provide the best overall performance.

1. INTRODUCTION

Many analysts expect streaming stored video to be the dominant traffic type in the Internet in the upcoming years. As with Web objects, video data can be transported to the client in many different ways, including (i) directly from origin server to client; (ii) through intermediate ISP caches; and (iii) through content distribution networks (CDNs) such as the Akamai network. In designing new strategies for distributing stored video over the Internet, we also must take into account that access to the Internet is highly heterogeneous [1, 2]. For this reason, video content providers typically provide multiple quality levels, with each quality level having a different encoding rate.

Multiple quality levels can be created by encoding video into multiple versions, each version encoded at a different rate. However, multiple versions of the same video can cause large increases in the amount of storage. Layered encoding (also known as hierarchical encoding) can also be used to create multiple quality levels. The storage requirements at a server for maintaining multiple layers is typically much less than maintaining the same number of versions. However, creating video layers generates additional bandwidth overhead [3, 4]. In particular, for the same quality level, layered encoding typically requires more transmission bandwidth than does a video version.

Given the presence of a caching and/or content distribution network infrastructure, and the need for multiple video quality levels, in this paper we compare distributing video versions to distributing video layers. We also examine mixed strategies consisting of both versions and layers. Broadly speaking, we find that mixed strategies that use both versions and layers provide the most robust performance.

1.1. Related Work

De Cuetos *et al* [5] and Kim *et al* [6] also compared streaming of video versions to streaming of video layers. However, they focused on time–dependent streaming of a single video from origin server to client; they did not take into account an intermediate cache sitting between origin servers and clients.

Kangasharju *et al* [7] considered caching strategies for layered video. However, they did not take into account multiple versions, and therefore did not compare caching layers, caching versions, and mixed strategies.

2. MODEL AND NOTATION

Fig. 1 illustrates our architecture for video caching. Suppose there are M videos available; and all of them are stored on the origin servers. Popular videos are cached in a proxy server, which is located close to its client community.

2.1. Proxy Server

The proxy server is connected to the origin servers via a wide area network (e.g., the Internet). We model the bandwidth available for streaming from the origin servers to the proxy server as a bottleneck link of fixed capacity C (bit/sec). The proxy is connected to the clients via a local access network, which could be a LAN running over Ethernet, or a residential access network using xDSL or HFC technologies. For the purposes of this study, we assume that there is abundant bandwidth for streaming from the proxy to the clients. We model the proxy server as having a storage capacity of G (bytes) and having infinite storage bandwidth (for reading from storage).

In this study each video can be encoded into either versions or layers. So, for the given proxy storage capacity G, link bandwidth C, video and request characteristics, our goal is to cache video layers and/or versions so as to maximize the number of streams that can be supported by the video caching system. We consider a caching strategy as optimal if given the bottleneck link C and the cache space G it maximizes the throughput, i.e., the long run rate at which video requests are satisfied. For versions, we suppose that there are two possible versions, namely, a high–quality version and a low–quality version. For layers, we suppose that the video is encoded into two layers, namely, a base layer and an enhancement layer. Thus, each video has four objects associated with it: a low–quality version, a high–quality version, a base layer, and an enhancement layer. We denote these four objects by l, h, b, and e, respectively.

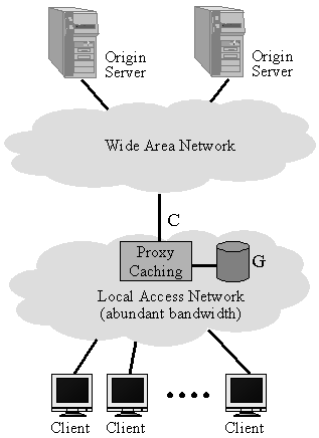

Figure 1: Architecture for adaptive video caching and streaming.

If $T(m)$ is the length of video m, $m = 1, \ldots, M$, in seconds and $r(m)$ is the encoding rate for one of the versions or layers, then the corresponding storage requirement for the object is $S(m) = T(m) \cdot r(m)$. We naturally assume that the rate of the high–quality version is greater than the rate of the low–quality version, i.e., $r_h(m) > r_l(m)$.

In order to compare the caching of layers and versions, we suppose throughout that the encodings are such that the visual quality of the base layer is the same as the visual quality of the low–quality version; and the video quality of the base and enhancement layer combined is the same as the high–quality version. However, due to encoding overhead to create layers, we do not assume that the layers and versions have the same rates. Instead, we make the following three natural *Rate Assumptions* which are based on video encoding experiments [3, 4]:

1. Due to the overhead of layered encoding, the base layer has at least the same rate as the low–quality version, i.e., $r_b(m) = r_l(m) \cdot [1 + O_l(m)]$ where $O_l(m) \geq 0$ is the low–quality coding overhead.
2. Again due to the overhead of layered encoding, the base and enhancement layers together have at least the same rate as the high–quality version, i.e., $r_b(m) + r_e(m) = r_h(m) \cdot [1 + O_h(m)]$ where $O_h(m) \geq 0$ is the high–quality coding overhead.
3. The base and enhancement layers together have smaller rate than the two versions, i.e., $r_b(m)+r_e(m) < r_l(m)+r_h(m)$.

For any video, the proxy can contain objects made from versions and/or layers. However, we assume the *decoding constraint*, namely, that the proxy never caches the enhancement layer if the base layer is not cached. When a request arrives to the proxy for some low–quality video, the proxy can satisfy the request if it is currently storing either the low–quality version or the base layer of the video. Otherwise, the proxy must obtain either the low–quality version or the base layer from the origin server and relay the object to the requesting client. When a request arrives to the proxy for some high–quality video, the proxy can satisfy the request if it is currently storing either the high–quality version or if it is storing both the base and enhancement layers of the video. Otherwise, it must retrieve an object from the network to satisfy the request. If the proxy has stored the base layer, then the proxy can retrieve either the enhancement layer or the high–quality version.

2.2. Basic Properties

For a given video, there are four cachable objects. Thus, there are $2^4 = 16$ different combinations of objects that can be put in the cache, including putting no object in the cache. This is a daunting number of combinations to analyze. Fortunately, without loss of generality, we may restrict ourselves to only five of the combinations:

Theorem 1 *There is an optimal caching configuration such that for each video one of the following five object combinations is used: $\emptyset$, $\{l\}$, $\{h\}$, $\{b\}$, or $\{b,\ e\}$. In other words, for each given video we either cache just the low–quality version, just the high–quality version, just the base layer, the base and enhancement layers together, or no objects at all.*

Proof: Due to the decoding constraint for layered video, we can rule out all combinations that include e but not b.

Now consider $\{b,\ h\}$. Note that Rate Assumptions 3 and 1 together imply that $r_h(m) > r_e(m)$. Hence $r_b(m) + r_h(m) > r_b(m) + r_e(m)$. It follows from this last expression that we can replace the combination $\{b,\ h\}$ with $\{b,\ e\}$ and use less storage while still satisfying all requests at the proxy for the video. Thus we can rule out $\{b,\ h\}$.

Now consider $\{b,\ l\}$, $\{b,\ l,\ e\}$, $\{b,\ l,\ h\}$, $\{b,\ l,\ h,\ e\}$. By caching the base layer, we satisfy all low–quality requests and we partially satisfy higher quality requests (only need to get enhancement layer from network). If we additionally cache the low–quality version, we take up more storage and we do not satisfy more requests for low–quality video. Combining this observation with $r_h(m) > r_e(m)$ implies that if we cache the base layer, then there is no need to also cache the low–quality layer. Thus we can rule out all these four cases.

Now consider $\{l,\ h\}$. This combination will satisfy all requests at the proxy. However, the combination $\{b,\ e\}$ also satisfies all requests and, by Rate Assumption 3, takes less storage. Thus, we can rule out $\{l,\ h\}$.

Finally, we can also rule out $\{b,\ e,\ h\}$ since the combination $\{b,\ e\}$ also satisfies all requests but takes less storage. ■

As a corollary to the above theorem, for any given video we use either versions or layers but not both.

Motivated by the above theorem, in the following sections we will propose and examine some strategies for caching layer and version objects. But it is also useful to make a few additional *Observations* about extreme cases:

1. For a given video if all (or "nearly all") requests are for the low–quality version (and none or "nearly none" are for the high–quality version), then we would either cache the low–quality version or cache no objects for that video, i.e., for object combination we would use either $\{l\}$ or $\emptyset$.
2. Similarly, if for a given video if all (or "nearly all") requests are for the high–quality version, we would use either $\{h\}$ or $\emptyset$.
3. If there is no overhead for layered encoding, that is, if $O_l(m) = O_h(m) = 0$, then for video m we would only use layers; in particular, we would use either $\emptyset$, $\{b\}$ or $\{b,\ e\}$.

However, when (i) there is layering overhead, and (ii) request rates for low– and high–quality versions are both significant, then it is not obvious whether we should use versions or layers; furthermore, for some videos it may be preferable to use versions whereas for others it may be preferable to use layers.

3. ANALYTICAL MODEL AND RESULTS

We start by modeling the steady–state cache performance. We assume that the request pattern is known *a priori* and does not change dynamically. Suppose that there are M videos and requests for video streams arrive according to a Poisson process with rate λ (requests/hour). Let j denote the requested quality level with $j = 0$ indicating a request for a low quality video, and $j = 1$ for a high quality video. Let $p(j,m),\ j = 0,\ 1;\ m = 1,\ldots,M$, denote the probability that a given request is for the j–quality stream of video m. As a proper mass distribution the $p(j,m)$'s satisfy $\sum_{m=1}^{M}\sum_{j=0}^{1} p(j,m) = 1$.

The corollary to Theorem 1 suggests three caching strategies, namely:

1. Pure version caching, where we cache only video versions.
2. Pure layer caching, where we cache only video layers.
3. Mixed caching, where we cache layers for some videos and versions for others.

For all three caching strategies we first order the request probabilities $p(j,m)$ in decreasing order. We then fill the cache by considering the objects (j,m) that are the most requested. First, we put the object (j,m) with the largest request probability $p(j,m)$ into the cache. Next, we cache the object (j,m) with the next largest probability $p(j,m)$, and so on. If at some point (as the cache fills up) the object needed to satisfy the request with the next largest request probability does not fit into the remaining cache space, we skip this object and try to cache the objects with the next largest request probabilities.

With *pure version caching* we cache the high quality version of video m if the next largest probability $p(j,m)$ is for the high quality stream of video m (i.e., $j = 1$). On the other hand, if the next largest probability is for the low quality stream of video m, then we cache the low quality version of video m.

With *pure layer caching* we cache the base layer of video m if the next largest request probability $p(j,m)$ is for low quality stream of video m. On the other hand, if the next largest probability is for the high quality stream of video m, then we cache both base and enhancement layer of video m. If the base layer has already been cached, i.e., if $p(0,m) > p(1,m)$, then we need to cache the enhancement layer only.

With *mixed caching* we cache the high quality version of video m if the next largest $p(j,m)$ is for the high quality stream of video m and no other object of the video has been cached. On the other hand, if the next largest probability is for the low quality stream of video m and no other object of the video has been cached, then we (i) cache the low version of video m if $r_b(m) > r_l(m)$, and (ii) cache the base layer of video m if $r_b(m) = r_l(m)$. However, if we have already cached the low (or high) quality version of a given video and the next largest probability is for a different quality of the video, then we replace the low (or high) quality version of the video with the base and enhancement layer of the video.

3.1. Video Caching Model

In this section we develop an analytical model for the caching and streaming of video layers and versions. We derive expressions for the blocking probability of a client request and the long run rate at which client requests are satisfied. To keep track of the objects in the cache we introduce a vector of cache indicators $\boldsymbol{c} = (c_1, c_2, \ldots, c_M)$, with $c_m = \{\emptyset\},\ \{l\},\ \{h\},\ \{l,\ h\},\ \{b\}$, or $\{b,\ e\}$, for $m = 1,\ldots,M$. c_m indicates whether no object, the low–quality version, the high–quality version, both the low– and high–quality version, the base layer, or the base layer together with the enhancement layer is cached for video m. In our model we focus on the bottleneck link of capacity C, that connects the proxy server to the origin servers. We model this link as a stochastic knapsack [8]. Let $b_{c_m}(j,m),\ j = 0,\ 1,\ m = 1,\ldots,M$, denote the link capacity required for satisfying a request for a j–quality stream of video m, given that the object(s) c_m are cached for video m. Let $\boldsymbol{b_c} = (b_{c_m}(j,m)),\ j = 0,\ 1,\ m = 1,\ldots,M$, be the vector of the bandwidth requirements of the requests. Let $\boldsymbol{n} = (n(j,m)),\ j = 0,\ 1,\ m = 1,\ldots,M$, be the vector of the numbers of ongoing j–quality streams of video m. Let $\mathcal{S_c} = \{\boldsymbol{n} : \boldsymbol{b_c}\cdot\boldsymbol{n} \le C\}$ be the state space of the stochastic knapsack model of the bottleneck link, where $\boldsymbol{b_c}\cdot\boldsymbol{n} = \sum_{m=1}^{M}\sum_{j=0}^{1} b_{c_m}(j,m)\cdot n(j,m)$. Furthermore, let $\mathcal{S_c}(j,m)$ be the subset of states in which the knapsack (i.e., the bottleneck link) admits a stream with the bandwidth requirement $b_{c_m}(j,m)$. We have $\mathcal{S_c}(j,m) = \{\boldsymbol{n} \in \mathcal{S_c} : \boldsymbol{b_c}\cdot\boldsymbol{n} \le C - b_{c_m}(j,m)\}$. The blocking probabilities can be explicitly expressed as

$$B_{\boldsymbol{c}}(j,m) = 1 - \frac{\sum_{\boldsymbol{n}\in\mathcal{S}_{\boldsymbol{c}}(j,m)}\prod_{m=1}^{M}\prod_{j=0}^{1}(\rho(j,m))^{n(j,m)}/(n(j,m))!}{\sum_{\boldsymbol{n}\in\mathcal{S}_{\boldsymbol{c}}}\prod_{m=1}^{M}\prod_{j=0}^{1}(\rho(j,m))^{n(j,m)}/(n(j,m))!}$$

where $\rho(j,m) = \lambda p(j,m)T(m)$ is the load offered by requests for j–quality streams of video m. These blocking probabilities can be efficiently calculated using the recursive Kaufman–Roberts algorithm [8, p. 23]. The expected blocking probability of a client's request is given by $B(\boldsymbol{c}) = \sum_{m=1}^{M}\sum_{j=0}^{1} p(j,m)B_{\boldsymbol{c}}(j,m)$. The long run throughput, i.e., the long run rate at which client requests are satisfied is given by

$$TH(\boldsymbol{c}) = \lambda\cdot\sum_{m=1}^{M}\sum_{j=0}^{1} p(j,m)(1 - B_{\boldsymbol{c}}(j,m)).$$

We define the normalized throughput $TH_n(\boldsymbol{c})$ as the ratio of the rate of satisfied requests to the total request arrival rate, i.e., $TH_n(\boldsymbol{c}) = TH(\boldsymbol{c})/\lambda$.

3.2. Numerical Results

We assume that there are M = 1,000 different videos. For a given video m we generate the version and layer rates as follows. The rate of the high quality version $r_h(m)$ is drawn randomly from a uniform distribution between 2 and 6 Mbps. The rate of the low quality version $r_l(m)$ is uniformly drawn between $0.5\cdot r_h(m)$ and $0.7\cdot r_h(m)$. The length of the video $T(m)$ is drawn from an exponential distribution with an average length of one hour.

We assume that the aggregate rate for the layered video has an overhead $O_h(m)$ over the high quality version, i.e., $r_b(m) + r_e(m) = [1 + O_h(m)]\cdot r_h(m)$. We consider two cases: (i) $r_b(m) = r_l(m)$, and (ii) $r_b(m) > r_l(m)$, in this case we vary $r_b(m)$ between $r_l(m)$ and $[1 + O_h(m)]\cdot r_l(m)$. With $r_b(m)$ fixed, the rate of the enhancement layer $r_e(m)$ is then computed as $r_e(m) = [1 + O_h(m)]\cdot r_h(m) - r_b(m)$.

Client requests arrive according to a Poisson process. The average request arrival rate is λ = 270 requests/hour. We set $p(0,m) = q\cdot p_m$ and $p(1,m) = (1-q)\cdot p_m$ for $m = 1,\ldots,M$, where q as a system parameter in our numerical analysis, and the p_m's are drawn from a Zipf distribution with parameter $\zeta = 1$.

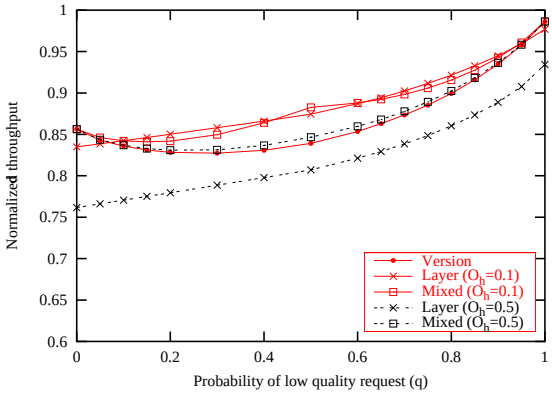

Figure 2: Results for varying probability of low quality requests ($r_b > r_l$).

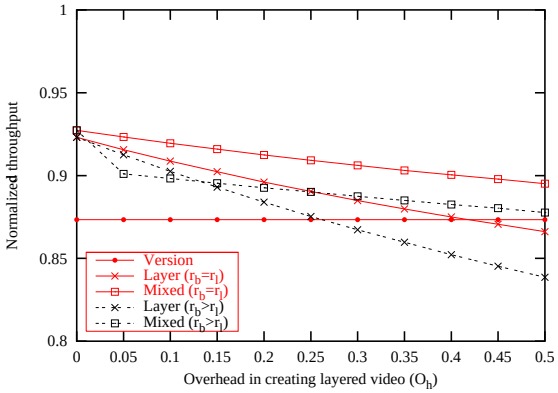

Figure 3: Results for varying amount of overhead of layered encoding ($q = 0.7$).

The cache size is set to G = 200 Gbytes and the link capacity is C = 150 Mbps.

In Fig. 2 we plot the normalized throughput as a function of the probability of a low quality request q. The results show that caching layers is favorable when the requests are non-homogeneous ($0.1 < q < 1$) and the overhead is low. We see that the throughput for pure layer caching increases monotonically as more requests are for low quality videos and decreases with increasing overhead. Pure version caching is only favorable in case of homogeneous request quality, i.e., all requests are either for high quality ($q = 0$) or for low quality streams ($q = 1$). The largest throughput is achieved if all requests are for low quality streams. This is expected because in this scenario more videos are cached and hence the cache hit rate is higher compared to a scenario where all requests are for high quality streams. The throughput is lowest when the requests are non–homogeneous as sometimes we need to cache both the low– and the high–quality version. The results indicate that mixed caching strikes a good balance between pure layer caching and pure version caching for all cases and offers the best overall performance.

Fig. 3 gives the normalized throughput as a function of the overhead O_h of layered encoding. We can clearly see that mixed caching gives better performance than pure version caching and pure layer caching for the range of overhead. Its performance is less sensitive to the overhead than pure layer caching. We have also found that the superiority of mixed caching is independent of the cache size, link capacity and request arrival rate; see [9] for a detailed study.

4. ADAPTIVE CACHING

While we have focused on a *steady state caching model* with *a priori* known request pattern so far in this paper, in the extended version [9] we also study extensively the *adaptive caching model*, where the request distribution is not known in advance and caching decisions are made on the fly. From this study we arrived at the following guidelines for distributing multi-quality video in the Internet:

1. Caches and CDN servers should be partially pre-filled with the most popular videos. If there are requests for both quality levels of a popular video, then the server should cache both the base and the enhancement layer of the video (rather than use versions). It is important to pre-fill the cache with the popular videos; otherwise, continuously streaming moderately-popular videos may prevent popular videos from getting stored in the cache.
2. For a first-time request of a video with unknown popularity, the origin server should stream the requested quality level as a *version*, and the proxy should not cache the version. If the video experiences multiple requests, then layers should be streamed and stored in the cache.
3. Although we should use versions to stream first-time requests from origin server to client, we should not cache versions (unless all the requests for a specific video are for one quality level).

5. REFERENCES

[1] T. Abdelzaher and N. Bhatti, "Web server qos management by adaptive content delivery," in *Proc. of Int. Workshop on QoS*, May 1999.

[2] W. Ma, I. Bedner, G. Chang, A. Kuchinsky, and H.J. Zhang, "A framework for adaptive content delivery in heterogeneous network environments," in *Proc. of MMCN 2000*, San Jose, CA, January 2000.

[3] K. Chandra and A. Reibman, "Modeling one- and two-layer variable bit rate video," *IEEE/ACM Trans. on Networking*, vol. 7, no. 3, pp. 398–413, June 1999.

[4] J. Kimura, F. Tobagi, J. Pulido, and P. Emstad, "Perceived quality and bandwidth characterization of layered mpeg-2 video encoding," in *SPIE Int. Symposium on Voice, Video and Data Communications*, Boston, MA, September 1999.

[5] P. De Cuetos, D. Saparilla, and K. Ross, "Adaptive streaming of stored video in a tcp-friendly context: Multiple versions or multiple layers?," in *Proc. of Int. Packet Video Workshop*, Kyongju, Korea, April 2001.

[6] T. Kim and M.H. Ammar, "A comparison of layering and stream replication video multicast schemes," in *Prox. of NOSSDAV 2001*, Port Jefferson, NY, Jun 2001.

[7] J. Kangasharju, F. Hartanto, M. Reisslein, and K. Ross, "Distributing layered encoded video through caches," in *Proc. of IEEE INFOCOM*, Anchorage, AL, April 2001.

[8] K.W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer-Verlag, 1995.

[9] F. Hartanto, J. Kangasharju, M. Reisslein, and K. Ross, "Caching Video Objects: Layers vs Versions," in *Tech. Rep.*, Dept. of Information Engineering, The Chinese University of Hong Kong, Jan. 2002, `http://www.ie.cuhk.edu.hk/~felix`. Also available at `http://www.eas.asu.edu/~mre`.