A Framework for Encoding and Caching of Video for Quality Adaptive Progressive Download

Siddhartha Chattopadhyay
Department of Computer Science
The University of Georgia
Athens, GA 30602-7404, USA
siddh@cs.uga.edu

Lakshmish Ramaswamy
Department of Computer Science
The University of Georgia
Athens, GA 30602-7404, USA
laks@cs.uga.edu

Suchendra M. Bhandarkar
Department of Computer Science
The University of Georgia
Athens, GA 30602-7404, USA
suchi@cs.uga.edu

ABSTRACT
Progressive download of multimedia objects over the Internet (e.g. www.youtube.com), where the video is downloaded and viewed during the download process, has become an increasingly popular alternative to multimedia streaming. Due to the fluctuating bandwidth and latency of the Internet, progressive download is often not fast enough, often resulting in intermittent stalling of the video. In this paper, we first propose a variation of the existing MPEG Fine Grained Scalability (FGS) profile to create a layered video representation that is suitable for progressive download in an environment characterized by varying bitrate. We also propose an efficient caching scheme that is specifically tailored for the proposed layered video representation. The proposed layered version of the Greedy-Dual-Size cache replacement policy is shown to reduce the latency observed by the client during progressive download of video in a varying bitrate environment. Experimental results demonstrate that the proposed caching scheme improves the latency of progressive video downloads as well as the server efficiency.

Categories and Subject Descriptors
C.2.4 Distributed Systems - Distributed applications

General Terms
Algorithms

Keywords
Layered Video Caching, Layered Greedy Dual Size, progressive download

1. INTRODUCTION
Progressive download of video over the Internet has become an increasingly popular alternative to multimedia streaming (e.g., YouTube [1]). Progressive download of video is similar to standard HTTP download of web content, except for the fact that video playback starts the moment a sufficient amount of video content has been downloaded. Most streaming servers can dynamically adapt the video bitrate to allow for continuous streaming in environments characterized by dynamically changing bandwidth. However, progressive download schemes often lack this bitrate adaptation capability since the video is typically encoded at a single bitrate. Moreover, caching of video for progressive download typically involves caching of the entire video such that the entire video is replaced during cache replacement. Thus, when the replaced video is requested by the client again, the entire video has to be fetched from the server, resulting in inefficient utilization of server and network resources. Although there is considerable published literature on caching of adaptive multimedia streams [2], [3], the techniques described therein cannot be directly applied to progressive download of video

In this paper, we propose a framework for quality adaptive progressive download (QAPD) of video. The proposed framework makes two major contributions:

- A novel layered video representation scheme, inspired by the MPEG Fine Grained Scalability (FGS) profile, which can be hosted simply on a standard HTTP web server.
- An efficient caching mechanism to significantly reduce the client-observed latency. In the proposed scheme, a simple proxy web server can perform the role of a proxy video server. Appropriate, quality metrics, that are specific to layered multimedia representation, are proposed to assess the performance of the proposed QAPD caching framework.

The proposed QAPD framework, with caching, provides considerable advantage over quality adaptive FGS (QAFGS) streaming. First, QAPD files can be hosted simply on a cost-effective web server with fewer CPU and memory requirements than those of a streaming server in case of QAFGS files. Second, the load on the simple web server in case of QAPD files is considerably lower than that on a dedicated streaming server in the case of QAFGS files. This is so because QAPD is driven mainly by simultaneous client requests for downloading various multimedia files. Finally, the QAPD proxy cache can also be hosted on a simple web server, whereas QAFGS requires the proxies to also have streaming capabilities.

2. QAPD
The proposed Quality Adaptive Progressive Download (QAPD) of video can be achieved by designing a layered representation of the video, which can be hosted on a HTTP server, followed by a method to enable variable bitrate progressive download.

2.1 Layered representation of video
Layered encoding for quality adaptive streaming is achieved by using the MPEG-FGS profile [4], [5], which partitions the video file into two layers; the base layer and the enhancement layer. The proposed layered representation scheme is an adaptation of the MPEG-FGS profile. Layered encoding of a video file V is achieved by first dividing V temporally into Groups of Pictures

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MM'07, September 23–28, 2007, Augsburg, Bavaria, Germany.
Copyright 2007 ACM 978-1-59593-701-8/07/0009...$5.00.

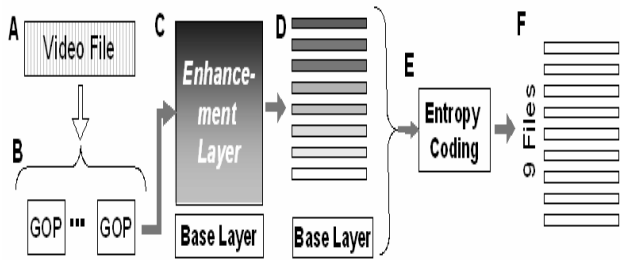

Figure 1. **The steps for creating layered video which can support quality adaptive progressive download**

(GOPs). For each GOP, the first frame is the I-frame; and the remaining frames are encoded as P-frames or B-frames using motion prediction [6]. Next, the DCT coefficients are computed for 8×8 blocks of the motion-predicted frames, followed by quantization of the coefficients, such that the quantized DCT coefficients can be represented using 8 bits. The base layer for the layered video representation can be generated by smoothing the frames in their respective color spaces, or equivalently, truncating the corresponding DCT coefficients [7]. The DCT-residue layer is computed by subtracting the truncated and quantized DCT coefficients of the base layer from the quantized DCT coefficients of the original frame. The DCT-residue layer is used to generate the enhancement layers. The first enhancement layer is generated by taking the most significant bit of the quantized residue of each DCT coefficient from the DCT-residue layer; the second layer generated by taking the second-most significant bit, and so on. In all, 8 enhancement layers can be generated from the DCT-residue layer, which, in conjunction with the base layer, constitute the best quality image. Finally, the base layer and the 8 enhancement layers are subject to lossless entropy coding (run length coding followed by arithmetic coding [6]) to generate the final, compressed set of 9 files, corresponding to a single multimedia file, as shown in Figure 1.

2.2 Layered multimedia-enabled QAPD

Using the above layered representation technique, a video file *f* is represented by 9 files, *f.0, f.1, f.2,..., f.8.* When a client sends a request for file *f* to the server, it essentially sends a request for the nine files *f.0,..., f.8.* The moment a GOP of the base layer is received, the client can commence video playback at the base quality. If the GOPs corresponding to the higher enhancement layers are also received, then a correspondingly higher quality video can be displayed simultaneously. Note that the proposed QAPD scheme entails parallel requests from the client which is in contrast to server-initiated variable bitrate streaming via dynamic rate adaptation, as is typically done in the case of MPEG-FGS. QAPD is thus essentially a client initiated technique.

3. LAYERED VIDEO CACHING

The success of progressive download is heavily dependent on ensuring low client-experienced latency. Thus it is desirable to cache portions of these files in the proximity of the client, so that the client can access the files quickly without waiting for the remote server. Note that traditional caching schemes cannot be used readily for QAPD as they are designed primarily for flat files. Consequently, we have designed a novel caching scheme for QAPD, termed as the Layered Multimedia Cache (LMC). The LMC acts as a proxy for the original server which hosts the media file denoted by *V*.

We assume that the LMC has a maximum storage capacity of *G* and is initially empty. A client request for the media file *V* is tantamount to a request for the base layer V_0 follwed by a request for each of the enhancement layers V_i $1 \le i \le 8$. Since none of these layers are in the LMC, the LMC requests the corresponding files from the main server, and caches them while relaying them to the client. We assume that, at a given point in time, the LMC at full capacity contains *M* media files, each consisting of a base layer file and 8 enhancement layer files. When trying to cache media file V_{M+1}, the LMC observes that the cache capacity *G* is exceeded. Obviously, some layers of some of the existing media files in the LMC need to be deleted in order to make space for the new media file V_{M+1}. This calls for an efficient cache replacement policy.

We propose an improvised version of the state-of-the-art Greedy Dual Size (GDS) cache replacement policy [8], [9] used commonly in web proxy caches. The improvised cache replacement scheme is termed as the Layered GDS (LGDS) cache replacement algorithm. The standard GDS algorithm is ineffective since it does not exploit the dependencies between the various layers of a media file. The proposed LGDS algorithm is described as follows. Each media file cached in the LMC can be viewed as a bin which contains the base layer and one or more enhancement layers corresponding to the media file. Each media file or bin is associated with a *retention-value* which quantifies the loss incurred by the LMC if some layers of that media file are deleted from the LMC. The lower the *retention-value*, the more dispensable the media file or its constituent layers. Since each media file consists of multiple layers, *the retention-value* of a media file (or bin) is that of its topmost layer which can be removed immediately. The *retention-value* of the j^{th} layer, $(0 \le j \le 8)$ of the i^{th} media file $V_{i,j}$ depends on three parameters: Latency(*i, j*), Size(*i, j*) and δPSNR(*i, j*). Latency(*i, j*) is the time taken to get $V_{i,j}$ from the server; Size(*i, j*) is the size (in bytes) of. $V_{i,j}$ and δPSNR(*i, j*) is the resulting change in PSNR when $V_{i,j}$ is added to the exisiting layers of the media file. Note that PSNR is a commonly used metric of visual quality of a video frame [10]. The *retention-value* of $V_{i,j}$ is denoted by RV(*i, j*) and is given by:

$$\text{RV}(i, j) = K \times \delta\text{PSNR}(i, j) \times \text{Latency}(i, j)/\text{Size}(i, j)$$

where *K* is a constant. The *retention-value* of a bin *i*, denoted by *retention-value(i),* is computed as

$$\textit{retention-value(i)} = \text{RV}(i, \text{TopLayer}(i)) \qquad (1)$$

where $0 \le \text{TopLayer}(i) \le 8$ is the topmost cached layer of the media file V_i. Note that the *retention-value* as computed in equation (1) is different from that used in the standard GDS algorithm, since the standard GDS algorithm does not consider the δPSNR value in its computation of the *retention-value.* Also note that the proposed LGDS algorithm, unlike the standard GDS algorithm, results in a caching scheme that is aware of the structural relationships amongst the various layers of a media file. The proposed LGDS algorithm uses the *retention-value*(*i*) of each bin *i* to decide which layer to cache and which layer to delete from the cache. When a media file or bin in the LMC scores a hit,

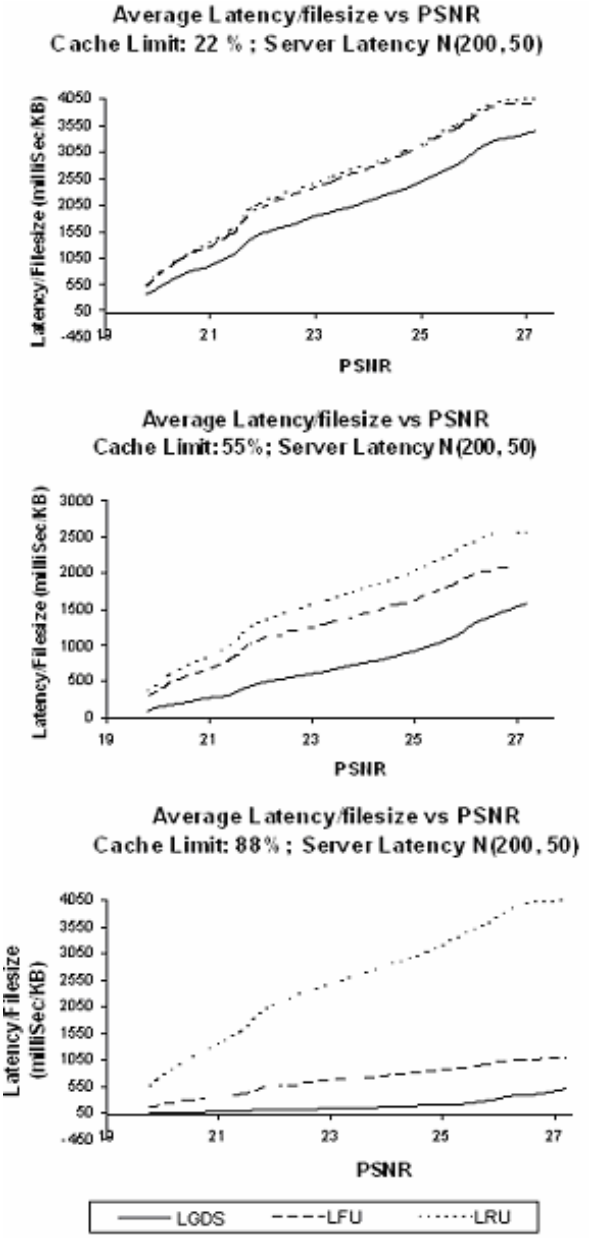

Figure 2. Plot of latency (in milliseconds) per KB of each layer, observed at the client end, compared to the average PSNR increase after each layer is added. Three cache sizes have been used: 22%, 55% and 88% of the total size of all the layers of all the files hosted in the server. The three lines correspond to three cache replacement policies implemented at the Layered-Multimedia-Cache. LGDS: Layered-Greedy-Dual-Size; LFU: Least Frequently Used, and LRU: Least Recently Used.

its *retention-value* reverts to its original value, i.e., the *retention value* assigned when the media was first saved in the cache.

Having described the working of the LMC, we now describe the methods used to evaluate its performance. It must be noted that layered multimedia caching differs from standard flat file caching, since each layer in the LMC is associated with a different visual quality enhancement factor. Thus, any scheme for LMC performance evaluation has to take this factor into account. In the following section, we describe the methodology and experiments for performance evaluation of the proposed LGDS algorithm in the context of the LMC followed by a discussion of the results.

4. EXPERIMENTS AND RESULTS

In this section, we present the experimental results of the evaluation of the efficiency of the proposed LMC, and the gains obtained by using the proposed LGDS cache replacement scheme.

4.1 LMC: Evaluation Methodology

In order to demonstrate the effectiveness of the proposed LGDS cache replacement policy for the LMC, we have compared the LGDS algorithm with two other popular cache replacement policies: Least Recently Used (LRU) and Least Frequently Used (LFU). Both, the LRU and LFU schemes were improvised to account for the structural dependency between the layers in the LMC. The LRU scheme is implemented as follows. For each bin, the time when the file was last accessed is recorded as the *retention-value* for that bin. When a layer is removed to make space for the new layer to be cached, the topmost layer is removed from the bin and the *retention-value* of the bin is left unchanged. When a video file (or bin) is accessed, the *retention-value* of the file (bin) is reassigned to the latest time of access (hit time). The LFU scheme is implemented in a similar manner by deleting the topmost layer from the selected bin based on the number of accesses recorded for that bin (media/video file).

For performance evaluation of the LMC we devise a metric which accounts for the change in visual quality of the media file as well as the latency incurred at the client end to receive a media file of that quality. First, we compute the layered representation of N media files. The N media files are assigned an arbitrary rank between 1 and N. The m^{th} layer of the n^{th} media file is denoted by $f(n, m)$, where $1 \le n \le N$, and $0 \le m \le 8$. R requests are made to the LMC using a Zipf distribution, which is known to emulate the access pattern for ranked files [11]. Requesting a file with rank k, from the client's point of view, is tantamount to requesting all the files $f(k, i)$, $0 \le i \le 8$ from the LMC. After each client request, the time taken to receive the i^{th} layer of the k^{th} file, $T'(k, i)$, $0 \le i \le 8$, and the resulting change in the PSNR value, $\delta PSNR(k, i)$, $0 \le i \le 8$, are noted. The mean of all the $T'(k, i)$ values, where $1 \le k \le N$ and $0 \le i \le 8$, is computed over all the R requests. After the R client requests have been serviced, the mean latency for layer j of the file with rank i is denoted by $T(i, j)$, $1 \le i \le N$, and $0 \le j \le 8$. Note that each of these layers corresponds to a physical file on the disc with size given by $Size(i, j)$. We normalize these latencies using the size information to get a metric $P'(i, j)$ given by

$$P'(i, j) = T(i, j)/\text{Size}(i, j)$$

$P'(i, j)$ is essentially the latency per byte of information for each layer of each file. Finally, the mean of $P'(i, j)$ is computed over all the N files, for each layer. The resulting metric $P(j)$, $0 \le j \le 8$, represents the average latency per byte for the j^{th} layer of the LMC. Similarly, the values of $\delta PSNR(i, j)$, $1 \le i \le N$, and $0 \le j \le 8$ are used to compute the mean cumulative PSNR, $PSNR(j)$, $0 \le j \le 8$, for each layer. $PSNR(j)$ serves as an objective evaluation of the visual quality of the files after the layer j is added to the layers 0,

1,.., j - 1. A plot of $P(j)$ versus PSNR(j), $0 \leq j \leq 8$, is used to assess the performance of the LMC.

The second performance metric is based on the cumulative sum S of the number of bytes transferred from the server during a cache miss, and the time T taken to send the media file to the client. The ratio, S/T, denotes the bandwidth efficiency of the server for a given cache replacement policy. A detailed discussion on the two metrics based on experimental results is given in the next subsection.

4.2 Results

Since trace data are not available for the proposed technique, we simulate network behavior using well known distributions. We create 20 layered video files (N = 20), each with one base layer and 8 enhancement layers. Each video is of 5 seconds duration, and with a GOP size of 15 and encoding efficiency of 1 frame per second. We simulate server latency by introducing a random delay modeled by a Gaussian distribution $\mathbf{N}(\mu, \sigma)$, where μ = 200 milliseconds, and σ = 50 milliseconds, in response to a request. We use cache sizes that are 22%, 55% and 88% of the sum of all the media file sizes to observe the effects of change in cache size. We send R = 1000 requests to the LMC, using a Zipf distribution with α = 0.9. For each value of cache size, we compute $P(j)$ and PSNR(j) as explained in the previous subsection.

Plots of $P(j)$ versus PSNR(j) for the three cache sizes are given in Figure 2. The plot essentially shows the latency per byte incurred while improving the quality of the media files. The lower the latency per byte incurred to improve the quality (i.e., increase PSNR value), the faster the client gets to view the better quality media file. In other words, lower latency per byte value for a given PSNR value signifies that better quality media can be progressively downloaded at a lower latency. From the plot, it is clear that the LMC using the proposed LGDS cache replacement policy outperforms both, the LFU and LRU cache replacement policies.

Figure 3 plots the second metric, i.e., server bandwidth efficiency, versus the cache size. The larger the server bandwidth efficiency, the better the server performance, and hence more efficient the cache is, in utilizing the server. From Figure 3, it is evident that the proposed LGDS cache replacement policy results in comparable, if not greater, server efficiency, and hence, a lighter server load, compared to the LFU and LRU cache replacement policies. Thus, with similar or lighter server load compared to the standard LFU and LRU cache replacement policies, the proposed LGDS cache replacement policy results in a significant reduction in the client-experieced latency during progressive download of the layered video.

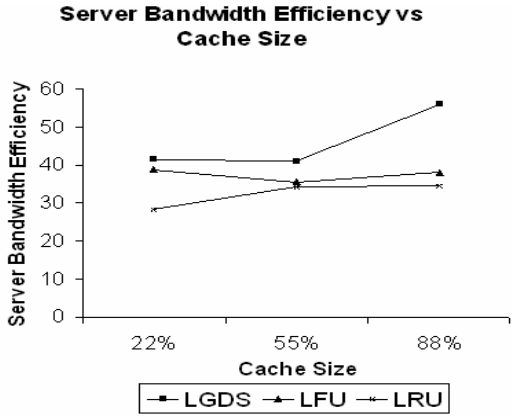

Figure 3. The server bandwidth efficiency (KB/s) versus the cache size after 1000 client requests.

5. CONCLUSIONS

In this paper, we have proposed a novel framework for quality adaptive progressive download (QAPD) of multimedia files. The proposed QAPD framework incorporates an efficient multi-layered video representation that is suitable for progressive video download at varying bitrates, an efficient Layered Multimedia Cache (LMC) in conjunction with a novel Layered Greedy Dual Size (LGDS) replacement policy, and novel evaluation metrics to quantify the performance of the QAPD system. Experimental comparisons with two other popular cache replacement policies, i.e., LRU and LFU, show that the proposed QAPD scheme, using the proposed LGDS cache replacement policy, significantly outperforms the conventional LFU and LRU replacement policies.

6. REFERENCES

[1] http://www.youtube.com

[2] R. Rejaie, H. Yu, M. Handley and D. Estrin, Multimedia proxy cache mechanism for quality adaptive streaming applications in the Internet, *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Apr. 2000, pp 980-989.

[3] J. Kangasharju, F. Hartanto, M. Reisslein and K. W. Ross, Distribution layered encoded video through caches, *Proc. IEEE INFOCOM,* Anchorage, AK, Apr. 2001, pp 1791 - 1800.

[4] W. Li, Overview of Fine Granularity Scalability in MPEG-4 Video Standard, *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, Mar. 2001, pp. 301-317.

[5] H. Radha, M. van der Schaar. and Y. Chen, The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP, *IEEE Trans. Multimedia,* vol. 3, Issue 1, Mar. 2001, pp. 53–68.

[6] I.E.G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*, Wiley, New York, NY, 2004.

[7] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

[8] S. Jin and A. Bestavros, Popularity-aware greedy dual-size web proxy caching algorithms, *Proc. 20th IEEE Intl. Conf. Distributed Computing Systems (ICDCS),* Taipei, Taiwan, Apr. 2000, pp. 254-261.

[9] P. Cao, and S. Irani, Cost Aware WWW Proxy Caching Algorithms*, Proc. USENIX Symp. Internet Technologies and Systems (USITS),* Monterey, CA, Dec. 1997, pp. 193-206.

[10] E. Davies, *Machine Vision: Theory, Algorithms and Practicalities,* Academic Press, San Diego, CA, 1990.

[11] P. Baldi, P. Frasconi, and P. Smyth, Modeling the Internet and the Web: Probabilistic Methods and Algorithms, Wiley, New York, NY, 2003.