MULTIMEDIA PROXY CACHING FOR VIDEO STREAMING APPLICATIONS.

Radhika R  
Dept. of Electrical Engineering, 
IISc, Bangalore.  
radhika@ee.iisc.ernet.in

Lawrence Jenkins  
Dept. of Electrical Engineering, 
IISc, Bangalore.  
lawrn@ee.iisc.ernet.in

ABSTRACT
This paper deals with the use of proxy servers to cache streaming video objects. In this paper, we describe a streaming architecture for layered video objects that uses the concept of layered caching. A new replacement policy is proposed, which is based on the playout time of a video object at the client. We define a new quality metric that takes the number of layers as well as the importance of each layer into consideration for quality measurements. Our simulation results compare the cache performance for different replacement policies and indicate the importance of playout time in the replacement scheme.

1. INTRODUCTION
The rapid growth of the Internet paralleled with the growth in digital multimedia and World Wide Web technology has increased the demand for real-time multimedia information on the web. The distribution of real-time audio and video over the Internet takes the form of streaming. Video streaming is resource intensive and as demand for video over the Internet increases, the network will not be able to handle the traffic, leading to scalability and latency problems. An elegant way to tackle such problems is through the use of proxy caches.

One of the main problems with streaming video is the client bandwidth heterogeneity. Scalable or layered coding has been suggested as a solution to the problems of client bandwidth heterogeneity, quality adaptation and congestion control [3], [4]. The load on the server and the latency in serving requests are important factors that affect the overall performance of the system. Proxy caching is an efficient method to overcome such problems in streaming video. There is a need to combine proxy caching and scalable encoding (for example, MPEG-2 layered coding or MPEG-4 fine grained scalability) in the form of layered proxy caching. The replacement policy used by the proxy is detrimental to the caching efficiency. The proxy may replace the individual layers of an object in a phased manner, i.e., fine-grained replacement or replace the whole object in its entirety, i.e., atomic replacement. In our framework, the replacement of objects in the cache is based on a weight assignment policy. We define a new weighting criterion which is based on the playout time of the video at the client end. Quality of the streams delivered from the cache is also an important aspect of proxy caching. To this end, we have defined a new metric to measure the quality of cached streams.

2. MULTIMEDIA PROXY CACHING FRAMEWORK
Fig. 1 shows our proxy caching framework. The proxy is located at the edge of the network\(^1\). There are two ways in which video can be streamed over the Internet: (i) By reserving bandwidth required by the application using resource reservation protocol (RSVP) [1], (ii) Through the use of a congestion controlled best effort service, where the service provided to the client depends on other traffic in the network. Our proxy caching framework pro-

\(^1\)We assume that the proxy server and the associated cache are a single unit.
poses to use the reserved bandwidth to fetch parts of the upper layers while storing only portions of the video clip which are played out at the client, in the cache.

The proxy first computes the number of layers that can be streamed with the available resources. If $N$ is the total number of layers into which the video is encoded, the proxy computes the number of layers $X$ that can be retrieved from the source with the reserved bandwidth, where $1 \leq X \leq N$. The proxy then checks the number of layers $L$ of the requested object that is already stored in the cache. The proxy then modifies the request by asking the server to stream layers $L+1$ through $X$. If the reservation module in RSVP cannot allocate enough bandwidth for at least one layer (i.e., $X=0$), then an error message is delivered to the client, and the next request is processed.

If there are enough resources to fetch all layers, then request all layers from the main server till the end of session.

If a part of the lower layers is already present in the cache, fetch the upper layers until the proxy detects that the length of the lower layers is not enough to serve the client request. The proxy stores information about the length of each cached layer, and when it senses that difference in the length of the upper layer and the next lower layer is $\delta$ (in seconds), it modifies the fetching strategy. The server now stops streaming the upper layer and sends the lower layers to serve the client request. Fig. 2 and Fig. 3 show two cases where the proxy attempts to improve the quality of the streams delivered to the clients. Choosing a value for $\delta$ is an important issue. We have chosen $\delta = 2$ times RTT (Round Trip Time between server and proxy) so that any modified request from the proxy to the server does not cause any breaks in the display of video.

3. REPLACEMENT POLICIES

The importance of each object in the cache is assessed by its weight which is determined by some property like frequency of access and recency of access. The replacement policy removes the objects in the cache in order of their weights, with more weighty objects being retained in the cache. We introduce a weight assignment policy that takes...
end-to-end “playout” time of an object into consideration. We define a new metric called the *playout ratio* that indicates the relative playout times of objects in the cache. The average playout time \( P_{\text{av}}(i) \) is calculated as

\[
\frac{\sum_{j=1}^{K} P_{i}(j)}{K}
\]

where \( K \) is the number of times object \( i \) has been requested, and \( P_{i}(j) \) is the playout time of \( i \)th object at its \( j \)th request. The playout ratio \( PR_{i} \) is given by:

\[
PR_{i} = \frac{P_{\text{av}}(i)}{T_{\text{tot}}(i)}
\]

where \( T_{\text{tot}}(i) \) is the total duration of the object \( i \). The weight of the \( i \)th object can now be defined as

\[
W_{i} = PR_{i}
\]

which will be henceforth referred to as PLAYOUT replacement policy. The measurement of the popularity should involve not only the number of times this video has been played out at the client but also the playout time at the client for each request. The frequency of occurrence is also important for calculation of the average playout time at the client across all sessions. Hence, we propose to combine LFU$^2$ weighting and PLAYOUT weighting to form a HYBRID weighting scheme.

The weight \( W_{i} \) of the \( i \)th object is given by

\[
W_{i} = LFU_{i}^{\alpha} \text{PLAYOUT}_{i}^{\beta}
\]

where \( LFU_{i} \) is the weight of the object assigned based on the frequency of access and \( \text{PLAYOUT}_{i} \) is the weight assigned by considering the average playout time of the object. Also, we choose \( \alpha + \beta = 1 \), \( 0 \leq \alpha \leq 1 \) and \( 0 \leq \beta \leq 1 \). \( \alpha \) and \( \beta \) are the relative weights of the two replacement schemes and hence chosen as normalized values that add to 1. The weights are chosen depending on the client interest in the video as indicated by the playout ratio. A high playout ratio implies that more weight needs to be assigned to the playout replacement policy which in turn implies a higher value for \( \beta \). We have chosen two different values of \( \alpha \) and \( \beta \) for simulation purposes. We assign \( \alpha = 0.7; \beta = 0.3 \) (HYBRID1 replacement scheme) and \( \alpha = 0.9; \beta = 0.1 \) (HYBRID2 replacement scheme).

### 3.1. Metrics used for cache performance measurement.

The most commonly used metrics to measure cache performance are hit ratio and byte hit ratio. We can define two types of cache hits to multimedia objects - a *partial hit* where only a part of the requested object is found in the cache and a *complete hit* where all the data requested by the client is available in the cache. Hit ratio calculation takes into account both partial and complete hits to objects. Byte hit ratio gives a better indication of performance, by measuring the total number of bytes requested and the total number of bytes served from the cache thereby indicating the usefulness of the cache in reducing server load.

We define a metric to measure the quality of the cached streams. One such metric is proposed in [2] but it may not indicate the relative perceptual importance of the layers of the cached streams, since it assigns equal weight to all layers of the object. We propose to take into account the relative perceptual significance of each layer the cached streams in quality evaluation. We achieve this by defining a new quality metric that assigns different weights to different layers. The *Quality Hit* gives a measure of the average quality of the object in the cache and is defined as follows:

\[
Q_{i} = \frac{\sum_{j=1}^{L_{i}} w_{j}}{N_{i}}
\]

where \( w_{j} \) is the weight of \( j \)th layer of object \( i \). \( L_{i} \) is the total number of layers of \( i \)th object in the cache and \( N_{i} \) is the total number of layers the object \( i \) is encoded into.

The numbering of the layers starts from the base layer, with the base layer being numbered 1, the next enhancement layer being numbered 2 and

---

\(^{2}\text{Least Frequently Used}\)
so on. The weight of layer $j$ of an object $i$ ($w_{ij}$) is
chosen as $\frac{1}{2^j}$ to capture the decreasing significance
of the higher layers compared to the lower layers.
The quality hit of an object lies between 0 and 1. The maximum quality hit ($Q_i = 1$) is achieved when all layers of the object are present in the cache.

4. SIMULATION RESULTS

In the absence of any real-time streaming system and a media trace file being available for reference, we have used a simulation based study to evaluate our work. The input to our system attempts to model a real-time streaming media server and the requests to objects stored in it. The input is generated by using the streaming media workload generator model described in [5]. The simulation studies consider a request for 2956 different objects with sizes that vary between 732 Kilobytes to 75544 Kilobytes. The total length of each video lies between 4 seconds and 393 seconds. The number of layers in each object is 4 and the stopping times are modeled according to the Pareto Distribution [5]. We start the simulation by assigning the cache size to be equal to 2.5% of the sum of the sizes of all objects. The replacement policy is invoked when the cache saturates and is unable to store any more objects. The simulation is run for different replacement policies with atomic and fine-grained replacement. The cache size is increased in steps and the simulations are re-run. Fig. 4, Fig. 5 and Fig. 6 show the plots of hit ratio, byte hit ratio and quality hit ratio for various replacement policies mentioned previously with atomic replacement policy. Fig. 7 and Fig. 8 show the byte hit ratio and quality hit ratio for different replacement policies with fine grained replacement. Our simulation results show that the HYBRID1 replacement scheme performs consistently well both in the case of atomic and fine grained replacement. Hence we can see that the playout time is an important parameter for replacement. Also byte hit ratio is a better measure for performance with fine grained replacement since hit ratio will consider partial hits.

5. CONCLUSION

In this paper, a framework for layered proxy caching system is introduced. The importance of the playout time of the video is also highlighted through two replacement policies, i.e., PLAYOUT and HYBRID. A new quality metric is introduced which considers the relative importance of each layer of a scalable encoded video.
REFERENCES


Figure 6: Quality Hit ratios for atomic replacement policies.

Figure 7: Byte Hit ratios for fine grained replacement policies.

Figure 8: Quality Hit ratios for fine grained replacement policies.