# Proxy Caching Mechanism for Multimedia Playback Streams in the Internet *

Reza Rejaie, Mark Handley, Haobo Yu, Deborah Estrin
University of Southern California
Department of Computer Science
Information Sciences Institute
Marina Del Rey, CA 90292
*reza,mjh,haoboy,estrin@isi.edu*

January 22, 1999

## Abstract

Despite the success of proxy caching in the Web, proxy servers have not been used effectively for caching of Internet multimedia streams such as audio and video. Explosive growth in demand for web-based streaming applications justifies the need for caching popular streams at a proxy server close to the interested clients.

Because of the need for congestion control in the Internet, multimedia streams should be quality adaptive. This implies that on a cache-hit, a proxy must replay a variable-quality cached stream whose quality is determined by the bandwidth of the first session.

This paper addresses the implications of congestion control and quality adaptation on proxy caching mechanisms. We present a fine-grain replacement algorithm for layered-encoded multimedia streams at Internet proxy servers, and describe a pre-fetching scheme to smooth out the variations in quality of a cached stream during subsequent playbacks. This enables the proxy to perform quality adaptation more effectively and maximizes the delivered quality. We also extend the semantics of *popularity* and introduce the idea of *weighted hit* to capture both the level of interest and the usefulness of a layer for a cached stream. Finally, we present our replacement algorithm and show that its interaction with prefetching results in the state of the cache converging to the optimal state such that the quality of a cached stream is proportional to its popularity, and the variations in quality of a cached stream are inversely proportional to its popularity. This implies that after serving several requests for a stream, the proxy can effectively hide low bandwidth paths to the original server from interested clients.

**Keywords:** Proxy Caching Mechanism, Congestion Control, Quality Adaptive Video Playback, Layered Transmission, Internet

## 1 Introduction

Proxy Caching has been a critical factor in providing large scale access to the web over the Internet. Today's web access patterns show frequent requests for a small number of popular objects at popular sites. Thus a popular object is transfered through the same network link once per request. In the absence of caching, this approach results in server overload, network congestion, higher latency, and even the possibility of rejection of a client's request. In essence, this results in *hot-spots* forming in the network. Caching proxies cure all these problems by distributing the load across the network.

During recent years, the rapid increase in commercial usage of the Internet has resulted in explosive growth in demand for web-based streaming applications [10, 13]. This trend is expected to continue, and justifies the need for caching popular streams at a proxy server close to the interested clients. Proxy caching for multimedia streams also provides a good opportunity to support VCR-functionalities more interactively as the control latencies to the proxy are lower.

Despite the success of proxy caching in the Web, proxy servers have not been used effectively for caching of Internet multimedia streams such as audio and video. There may be several reasons for this:

- Realtime streams such as video are several orders of magnitude larger than normal web objects. This may decrease their chance for being cached based on the current replacement algorithms.
- The number of embedded multimedia streams on the Web has been fairly limited. Moreover, access patterns to these streams are not well known.

*This work was supported by DARPA under contract No. DABT63-95-C0095 and DABT63-96-C-0054 as part of SPT and VINT projects

- Lack of an open, well-accepted and well-behaved transport protocol for streaming applications.

Realtime streams have several inherent properties that differentiate them from other web objects such as text or images. These properties can be exploited in the design of an effective caching mechanism:

- In contrast to other web objects, these streams do not require to be delivered at once. Instead, the server usually pipelines the data to the client through the network.
- Multimedia streams are able to change their size by adjusting their quality.

The main challenge for proxy caching of Internet multimedia streams is the need for congestion control. In the Internet all end-systems are expected to perform congestion control to keep the network utilization high while limiting overload and improving inter-protocol fairness[6]. Since streaming media flows must co-exist with TCP-based flows such as HTTP, this congestion control will need to be TCP-friendly[14]. To prevent receiver buffer underflow when available bandwidth is limited, while maximizing the quality to the end-user when additional bandwidth is available, multimedia streaming applications should be *quality adaptive* - that is, they should change the compression ratio of the streamed data according to available network bandwidth. Thus proxy caches are likely to cache data that depends on the available bandwidth to the first client that retrieved a multimedia stream.

Once a stream is cached, the proxy can replay it from the cache for subsequent requests but it still needs to perform congestion control and quality adaptation during delivery. However, using the variable-quality cached stream to perform quality adaptation for subsequent requests is problematic because there is no correlation between the variation in quality of the cached stream and the required quality for the new session.

We have been working on an end-to-end client/server architecture for playback of quality adaptive multimedia streams in a TCP-friendly fashion over the Internet[17]. There are many ways to adjust the quality but the one we are investigating is to use layered encoding. With a layered codec, the compressed data is spilt into a base layer which contains the most essential low quality information and other layers which provide optional enhancement information. Thus the more layers are played back, the better the quality becomes.

Layered organization of streams provides a perfect opportunity for a proxy cache to cope with variations in quality of a cached stream. If the cached stream is structured properly, the proxy is able to cope with variations in quality during subsequent playback by only playing a subset of layers from the cache, or by simultaneous playback from the cache and fetching additional layers from the original server.

### 1.1 Contribution of this Paper

This paper addresses the implications of congestion control on proxy caching mechanisms. Additionally, we present a fine-grain replacement algorithm for layered-encoded multimedia streams at Internet proxy servers, and describe a pre-fetching scheme to smooth out the variations in quality of a cached stream during subsequent playbacks. Thus it enables the proxy to perform quality adaptation more effectively and maximizes the delivered quality. We also extend the semantics of *popularity* and introduce the idea of *weighted hit* to capture both the level of interest and the usefulness of a layer for a cached stream. Finally, we present our replacement algorithm and show that its interaction with pre-fetching results in the state of the cache converging to the optimal state such that the quality of a cached stream is proportional to its popularity, and the variations in quality of a cached stream are inversely proportional to its popularity. This implies that after serving several requests for a stream, the proxy can effectively hide low bandwidth paths to the original server from interested clients. Thus the delivered quality of a popular stream is not limited to the available bandwidth from the original server. Instead, the quality of each stream is determined by their popularity and the average bandwidth between the proxy and interested clients.

The rest of this paper is organized as follows: First we present our end-to-end architecture and describe how proxy servers complement this architecture in section 2. The delivery procedures on a cache-miss and cache-hit are sketched out in section 3 where we describe a pre-fetching mechanism to cope with missing packets and variations in quality. We present details of the replacement algorithm in section 4. Section 5 addresses some of the related work on proxy caching for multimedia streams. Finally, section 6 concludes the paper and presents our future directions.

## 2 The Architecture

Fig. 1 depicts our proposed end-to-end architecture for playback of quality adaptive realtime streams in the Internet[17]. All streams are layered encoded and stored at the server's archive. Here we assume *linear-layered* encoding where all layers have the same bandwidth just for the sake of simplicity, but the architecture and the caching scheme can be extended for other bandwidth distribution of layered encoding as well.

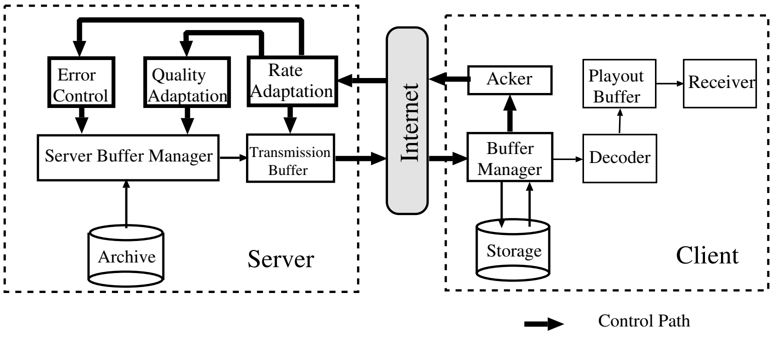


Figure 1: End-to-end architecture for realtime playback applications in the Internet

End-to-end congestion control is performed by the *Rate Adaptation Protocol (RAP)*[16] and *acker* modules at the server and client respectively. The RAP module continuously monitors the connection and regulates the server's transmission rate by controlling the inter-packet timing. The acker module acknowledges each packet, providing end-to-end feedback for monitoring the connection.

Error control is performed by the *error control* (EC) module at the server. It receives information about available bandwidth, loss rate and recent playout time from the RAP module. Based on this information, it either flushes packets from the server's buffer manager that were acknowledged or whose playout time has passed, or schedules retransmission of a lost packet if it has sufficiently high priority. The error control module can *selectively* retransmit those packets that have higher priority such as those from the base layer. Thus some losses are never retransmitted because they will miss their playout times, or they are low priority such as losses from the top layer.

The quality of the transmitted stream is adjusted by the *Quality Adaptation* (QA) module at the server. This module periodically obtains information about the available bandwidth from the RAP module. Combining this information with the average retransmission rate provided by the error control module, the QA module adjusts the quality of the transmitted stream by *adding or dropping layers* accordingly. The client usually buffers some data by slightly delaying the playback of the first packet. Rate adaptation happens on a timescale of round-trip times but layers are added and dropped on a longer timescale. A combination of receiver buffering and quality adaptation is able to cope with variations of available bandwidth. Short term variations of transmission rate can be absorbed by buffering without adding or dropping layers whereas long term changes in available bandwidth trigger the quality adaptation mechanism to adjust the delivered quality of the stream by changing the number of transmitted layers.

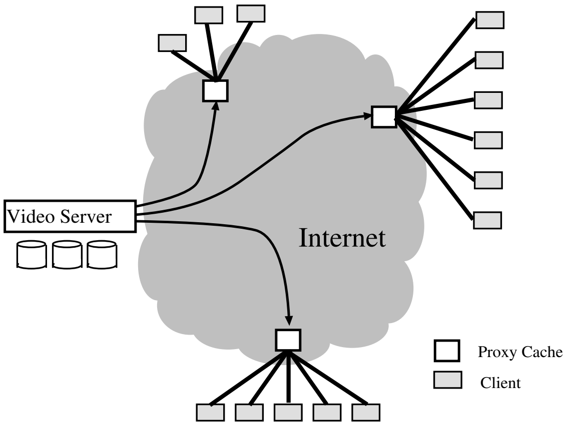


Figure 2: A typical scenario for a sever with heterogeneous clients

Proxy caches perfectly complement our end-to-end architecture. Fig. 2 depicts the whole architecture where a continuous media server plays back video or audio streams for heterogeneous clients upon their requests through a corresponding proxy cache. The server must be able to support a large number of requests simultaneously. Streams are usually large in size, so pre-fetching the entire stream and playing back from the local disk is not an option because of the large startup delay. In addition, the client may not have enough storage space to cache the entire stream. However, each client has sufficient buffer space to absorb short-term variations in bandwidth. Traffic is always routed through a corresponding proxy server that is associated with a group of clients in its vicinity. Thus the proxy is able to intercept each stream and cache it. To allow the proxy to perform fine-grain cache replacement, each layer of the encoded stream is divided into equal-size pieces called *segments*. Replacement is performed in the granularity of a segment. Each segment can be as small as a single packet or as big as several minutes of a stream. Having small segments prevents the cache space becoming fragmented while large segments require less coordination overhead. For a particular segment length, each segment is uniquely identified by the playout time of the first sample in that segment.

The amount of available storage space of the proxy servers is proportional to the number of their clients and it must be generally sufficient to cache a few popular streams. All streams between the original source and the client or the proxy server and the client must perform TCP-friendly congestion control and quality adaptation. This implies not only the original server but also the proxy server must be able to support congestion control and quality adaptation. We do not make any assumption about the inter-cache architecture. Our work is orthogonal and applicable to different inter-cache architectures [2, 21, 25].

Our goals are to:

- Maximize quality of the delivered stream to the clients.
- Minimize load on the server.
- Minimize startup latency.
- Provide low-latency VCR-functionality for the clients.

# 3 Delivery Procedure

This section describes the steps for delivery of a requested stream. We present a pre-fetching scheme that copes with variation in quality of the cached stream and enables the proxy server to effectively perform quality adaptation in different potential scenarios.

Each client directs its request to its configured proxy server. Upon arrival of a request, the proxy server looks up the cache for availability of the desired stream. The subsequent steps of the delivery procedure vary for a *cache miss* and a *cache hit* cases as we describe in the next two subsections.

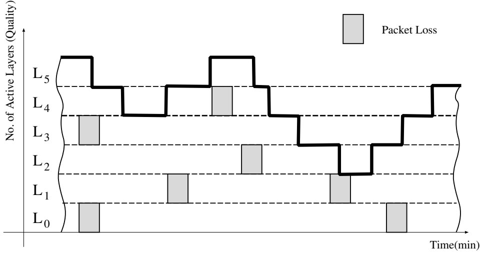


Figure 3: A delivered quality adaptive stream to the client

## 3.1 Relaying on a Cache Miss

On a cache miss, the request is relayed to the stream's original server[1] called source. Delivery of the stream is initiated from the source to the client through the proxy cache.

There are two options for how the proxy relays the stream:

- It may be completely transparent, passing data through from server to client and ACKs through from client to server, while caching the data as it passes.
- It may terminate the server-to-proxy RAP connection, sending its own ACKs, storing the stream in the cache, and maintain a separate RAP connection from the cache to the client.

The choice of strategy is transparent from the point of view of the source or the client. The former strategy is much simpler to implement at the cache, as the latter requires accurate prediction of which layers will be needed between the proxy and the client ahead of the time when we know the available bandwidth. If there is much less available bandwidth between the source and the proxy than between the proxy and the client, then this is not a difficult problem because we can assume that everything that reaches the proxy can also reach the client before its playout time, but in other circumstances this assumption does not hold. However, there is potential for improved performance by doing this as each of the two feedback loops (server-proxy and proxy-client) is shorter and has a lower loss rate than the concatenated server-proxy-client path. For simplicity, we shall assume that we use the former strategy, but the latter merits future research.

The source performs congestion control and quality adaptation based on the state of the session between the source and the client. Thus the quality of the delivered stream is determined by the average available bandwidth during the session. If the source is located behind a low-bandwidth path across the Internet, the average quality of the delivered stream is low. Furthermore, there may still be packets missing from the delivered stream that have not been repaired during the session due to a failure to successfully retransmit them before their playout time at the client. Fig. 3 shows an example portion of a stream that was cached at a proxy.

In summary, on a cache miss scenario, the client does not observe any benefit, in terms of startup latency or improvement in quality, from presence of the proxy cache. Thus the quality of the session on a cache miss is the same as a direct playback from the source to the client.

[1] With an appropriate inter-cache architecture, the request might be directed to another cache, but this doesn't change the basic mechanism

A missing stream is always cached during its first playback. If cache space is exhausted, a sufficient number of segments from another cached stream are selected for replacement, and are flushed to make room for the new stream. Details of the replacement algorithm are explained in section 4.

Once the proxy has cached a stream for the first time, it now has the option to attempt to fill in the remaining packet losses in the cached copy of the stream. It may wait until the next client requests playback, or it may alternatively pro-actively pre-fetch the missing packets from the source. If the proxy chooses to pre-fetch during an idle time, it can simply use TCP for delivery as there are no timing constrains, but if it fetches on demand, it will need to use RAP and combine this with the mechanism described below. Fig. 4 depicts a portion of a cached stream after repairing all the losses.

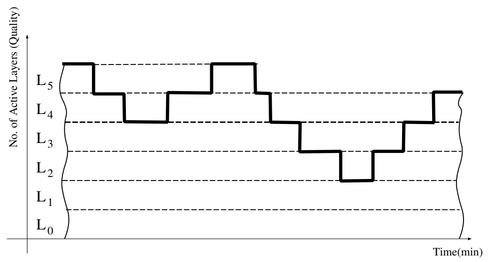


Figure 4: Part of a cached stream after repair of losses

## 3.2 Pre-fetching on a Cache Hit

On a cache hit, the proxy server immediately initiates playback of the requested stream from the available copy in the cache. Thus the client observes a substantially shorter startup latency compared with the cache miss case. The proxy behaves similarly to the original server and performs congestion control and quality adaptation. However, the connection between the proxy and the client is likely to have different bandwidth and round-trip-time characteristics. During each interval of the session two scenarios are possible:

1. $Playback_{avgBw} \leq Stored_{avgBw}$
2. $Playback_{avgBw} > Stored_{avgBw}$

where $Playback_{avgBw}$ and $Stored_{avgBw}$ denote average bandwidth of the playback session and the cached stream respectively. Note that during a complete session, the proxy may sequentially experience both of the above scenarios. We address each one of these scenarios separately.

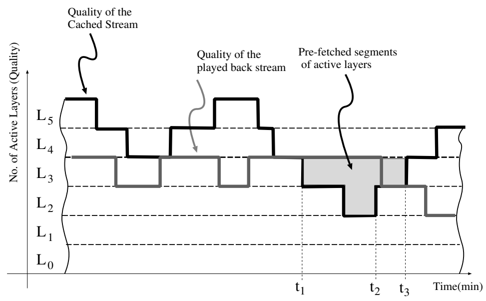


Figure 5: Delivery of lower bandwidth stream from the cache

### 3.2.1 $Playback_{avgBw} \leq Stored_{avgBw}$

In this scenario, the average quality(i.e. bandwidth) of the cached stream is adequate to perform quality adaptation for the current session. However, missing segments of the active layers of the cached streams could degrade efficiency of the quality adaptation mechanism. By active layers we refer to those layers of the cached stream that are likely to be transmitted by the quality adaptation mechanism. An example scenario is shown in fig.5. In this example the quality adaptation mechanism transmits up to four layers and the cached stream has up to six layers. But missing segments of $L_3$ during interval $[t_1, t_3]$ constrain the quality adaptation to transmit lower quality stream(i.e. 3 layers) despite availability of bandwidth between the cache and the client.

To prevent degradation of quality, the proxy should pre-fetch missing segments of the active layers before their playout time. Fig. 5 illustrates pre-fetched segments of $L_2$ and $L_3$ of the cached stream.

All the pre-fetched segments are cached even if they arrive later than their playout time. By pre-fetching the missing segments, the proxy smoothes out variations in quality of the cached stream across the active layers. The more a particular stream is played back from the cache, the smoother its quality becomes across the active layers. This decreases the probability of pre-fetching on subsequent cache hits for sessions with the same or lower average bandwidth. Fig 6 shows the quality of the cached stream after the first playback.

### 3.2.2 $Playback_{avgBw} \geq Stored_{avgBw}$

In this case, the proxy not only needs to pre-fetch the missing segments of the active layers but it also requires to pre-fetch higher quality layers whenever quality adaptation mechanism demands. The likelihood of observing missing segments depends on the frequency of access to the cached stream and the average bandwidth of the previous sessions.

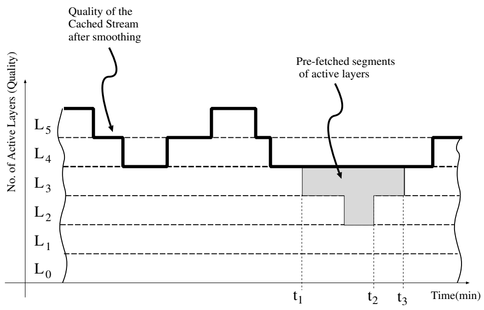


Figure 6: Quality of the cached stream after the smoothing

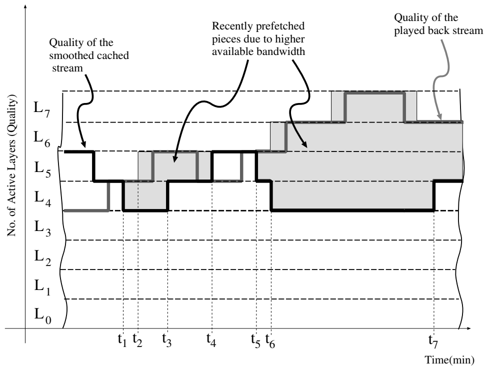


Figure 7: Delivery of higher bandwidth stream from the cache

The proxy needs to pre-fetch a higher layer as soon as the quality adaptation mechanism detects availability of higher bandwidth to improve the quality in the near future. This means that lower layers are played back from the cache while the higher layer is pre-fetched by the proxy ahead of time and is transmitted whenever quality adaptation mechanism adds a new layer. Once a new layer is added, it is continuously pre-fetched until the available proxy-client bandwidth dictates the layer is dropped or the session ends. Fig. 7 demonstrates pre-fetching of missing segments and new layers during a session. For example missing segments for $L_4$ and $L_5$ are pre-fetched during the interval $[t_1, t_3]$ and $[t_2, t_4]$. Whereas $L_6$ and $L_7$ were pre-fetched as added layers. Pre-fetched layers are always cached. Fig. 8 shows quality of the cached stream after a session with higher average bandwidth.

### 3.3 Issues and Trade-offs

As we mentioned earlier, the quality adaptation mechanism, adjusts the number of active layers based on long-term changes in available bandwidth. Although the number of active layers changes slowly, the time of the next adjustment is not known a priori because of the random changes

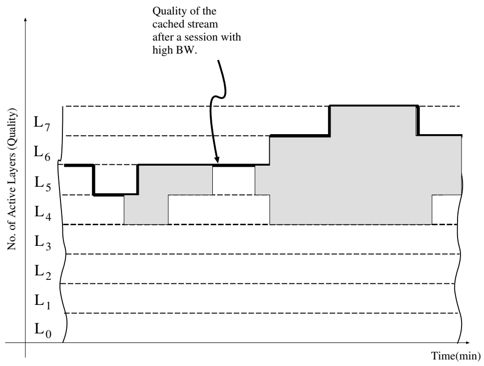


Figure 8: Quality of the cached stream after a session with higher avg. BW.

in available bandwidth. In other words, the quality adaptation mechanism can only estimate the number of active layers for the near future. This results in a trade-off for pre-fetching of missing segments or added layers. The earlier a required segment(either a missing segment or segment of a new layer) is pre-fetched, the higher the probability is that the segment will arrive before its playout time, but the less accurate is the prediction for requiring that segment. This means that a conservative approach results in better quality but is less efficient. Pre-fetching is illustrated in fig. 7. Note that when the prediction is incorrect, a few pre-fetched segments for $L_5$, $L_6$ or $L_7$ are not transmitted to the client.

Although all the pre-fetched segments during a session may not be played back, they are cached in the hope they will be used for the next request. This supports the conservative approach. To ensure in-time delivery of requested segments, the proxy should continuously monitor its round-trip-time to the active servers. This enables the proxy to issue a request ahead of time accordingly to absorb the delay in delivery. Pre-fetching is performed through a RAP connection because the strictly reliable stream-based nature of TCP may result in segments arriving after their playout point.

It is worth emphasizing that the more a particular cached stream is played back, the smoother its quality becomes across the active layers. Since the number of active layers is directly determined by the average bandwidth between the proxy and interested clients, this implies that the quality of the cached stream converges to the average quality across recent playback sessions.

# 4 Caching Mechanism

This section addresses different aspects of the caching mechanism for multimedia streams.

## 4.1 Replacement Issues

Current replacement algorithms usually make a binary decision on the caching of an atomic object, i.e. the object is cached or flushed in its entirety based on the popularity of the object. As we mentioned earlier, layered multimedia streams are able to adjust their quality. This allows the replacement algorithm to make a multi-valued decision for caching of multimedia streams. As the popularity of a stream decreases, its quality and consequently its size is reduced before it is finally flushed from the cache. In other words, popularity of a cached stream primarily affects its quality and then its status of residency in the cache.[2] The coarse-grain adjustment in quality is achieved by dropping the highest layer, called a *victim* layer. However, to maximize the efficiency of the cache and avoid fragmentation of the cache space, the victim layer is dropped with the granularity of a segment. To hide the startup latency, it is prefered to keep the initial segments of a layer in the cache. Furthermore, to minimize the variation in quality it is prefered to keep contiguous segments of a layer. This leads us to the replacement pattern in fig. 9. Once the highest layer is selected as a victim, its cached segments are flushed from the end toward the beginning in a demand-driven fashion.

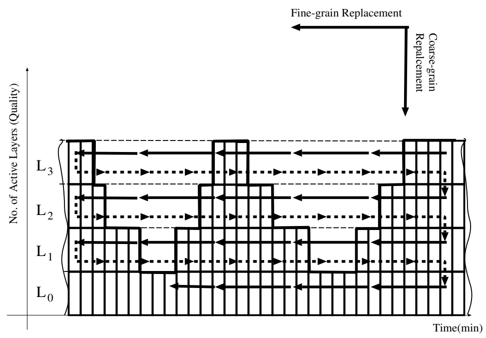


Figure 9: Pattern of flushing for a cached stream

## 4.2 Popularity Function

The chief goal of the replacement algorithm is for the state of the cache to converge to the optimal state after several playbacks. The cache state is optimal when the following conditions are met:

- The average quality of a cached stream is directly proportional to its popularity. Furthermore the average quality of the stream must converge to the average bandwidth across the most recent playback for interested clients.
- Variation of a cached stream is inversely proportional to its popularity.

We use a modified *hit ratio* of a cached stream as a metric to measure its popularity. The proxy can easily count number of cache hits for every cache resident stream, but we realize that the current definition of a *hit* is inadequate to effectively capture the subtleties of level of interest in a particular stream. This is mainly due to the time scale for delivery of multimedia streams being several orders of magnitude longer than for web objects. Most of the current schemes assign a binary value to a hit, i.e. 0 for lack of interest and 1 for each request. This model perfectly suites the atomic nature of interest in web objects, i,e, the client is either interested in the entire object or is not interested at all. In the context of streaming applications, the client can interact with the server and perform VCR-functionalities(i.e. Stop, Fwd, Rwd, Resume the session). Intuitively, the popularity of each stream must reflect the level of interest that is observed through this interaction. We assume that total duration of playback for each stream indicates the level of interest in that stream. For example if a client only watches half of one stream, their level of interest is half of a client who watches the entire display. This approach can also include weighted duration of fast forward or rewind with proper weighting.

Based on this observation we extended the semantic of a hit and introduce a *weighed hit*, called a *whit*, which is defined as follows:

$$whit = \frac{PlaybackTime}{StreamLength}, \quad 0 \leq whit \leq 1$$

where $PlaybackTime$ and $StreamLength$ denote total playback time and length of the entire stream respectively. The proxy server keeps track weighed hits on a per-layer basis. For each layer of a session, the total playback time for each layer is recorded and used to calculate the $whit$ for that layer at the end of the session. The cumulative value of $whit$ during a recent window is used as a popularity index of a layer of a cached stream. The popularity of each layer is recalculated at the end of a session as follows:

$$P = \sum_{x=t-\Delta}^{t} whit(x)$$

where $P$ and $\Delta$ denote popularity and the width of the popularity window respectively.

Notice that the behavior of the quality adaptation mechanism also affects the value of a cached *layer*. Since the available bandwidth directly controls number of active layers, the longer a layer is played back for interested clients

[2]This idea is also applicable to images since they can also be layered-encoded.

during recent sessions, the higher is the probability of using that layer in future sessions. For example, if the majority of clients interested in streaming $Titanic$ have low bandwidth, the quality adaptation mechanism can only play back say 3 layers. Thus higher layers should not be cached because there is a low probability of using those layers. This implies that total playback duration of a layer indicates its value. To incorporate this parameter, the server must separately calculate the popularity for each layer of a cached stream at the end of a session. Applying the definition of popularity on a per-layer basis is in fact compatible with our proposed replacement scheme because layered decoding guarantees that popularity of different layers of each stream monotonically decreases with the layer number[3].

The proxy maintains a popularity table such as table 1. Each table entry consists of stream name, layer number, popularity of the layer and a locking flag. Once the cache space is exhausted, the proxy flushes the last segments of the least popular layer(e.g. $L_2$ of "*The English Patient*" until sufficient space becomes available. Because of the decoding requirement, the least popular layer is always the highest layer of a stream. Popularity of this layer could be low due to lack of interest among clients, or lack of sufficient bandwidth to play this layer for interested clients, or both[4]. Note that except for the base layer of each stream, all segments of other layers can be flushed out if they are the least popular layer. The first few segments of the base layer for each stream are kept in the cache as long as its popularity is beyond a threshold to hide the startup latency of possible future request.

| $P$ | Lock | Stream name | Layer no. |
|---|---|---|---|
| 5.85 | 1 | $Titanic$ | $L_0$ |
| 4.92 | 1 | $Titanic$ | $L_1$ |
| 4.76 | 0 | $The\ English\ Patient$ | $L_0$ |
| 3.70 | 1 | $Titanic$ | $L_2$ |
| 3.50 | 0 | $Jerry\ Maguire$ | $L_0$ |
| 3.33 | 0 | *Apollo* 13 | $L_0$ |
| 2.30 | 0 | $Titanic$ | $L_3$ |
| 1.28 | 0 | $The\ English\ Patient$ | $L_1$ |

Table1: Sample of a popularity table

### 4.3 Thrashing

As we mentioned earlier, replacement for a web object is an atomic operation; the least popular objects are flushed and the new object is cached. However, in the context of multimedia streams, replacement is a timely process that proceeds gradually as the session continues. This causes the potential for thrashing where the tail of the highest layer of the cached stream (e.g. $L_3$) is flushed to make room for initial segments of a higher layer (e.g. $L_4$) of the same stream during a playback session with higher bandwidth. To avoid this, while a particular stream is played back from the cache, its layers are locked in the cache and can not be replaced. In practice, if a layer has not been played at all by this client then it need not be locked. But if a layer is ever played by a client then it is locked until the end of that session. At the end of the session, the weighted hit is calculated and consequently the popularity value of the that stream is updated in the popularity table. Then all the locked layers of that stream are unlocked.

## 5 Related Work

Memory caching for multimedia streams has been extensively studied in the context of multimedia servers [3, 5, 4, 9, 12]. The idea is to reduce disk access by grouping requests and retrieving a single stream for the entire group. In an abstract view, the problem is to design an object management mechanism that minimizes migration of objects among different levels of hierarchical storage system, i.e. tertiary and disk, disk and memory [7, 8]. Most of these studies have focused on resource management issues. Notice that there is an analogy between (tertiary,disk,memory) and (server, proxy, client). Object migrations among different levels of a hierarchical storage system is somewhat similar to object migration among server, proxy, and client. However, there is a fundamental difference between these two problems. The available bandwidth between levels of a hierarchical storage system is fixed and known a priori. In contrast, available bandwidth between server and proxy or proxy and client randomly changes with time. As a result applying the proposed resource management schemes across the network becomes problematic. Once these entities are scattered across a best-effort network, all connection must be congestion-controlled. This issue has not been previously addressed.

Video streams exhibit burstiness due to encoding algorithm and variations within and between frames. The variation poses a problem to both bandwidth requirement and playback buffer management. In order to smooth the burstiness during transmission, a number of schemes have been proposed. One scheme [22] prefetches and stores portions of a stream in proxy servers, and later use them to smooth the stream during playback. Another work [19] stores prefixes of multimedia streams in proxy caches. It's able to improve startup latency in addition to performing smooth-

[3]The decoding constraint requires that to decode a segment of layer $i$, corresponding segments for all the lower layers must be available

[4]Note that these three scenarios can be easily recognized from the distribution of popularity values among layers. Close popularity value imply lack of clients interest whereas widely variable popularity values imply lack of available bandwidth to play less popular layers.

ing.

Our work is complementary to the works on smoothing. They do not address congestion control of multimedia streams in the Internet, while we focus on design a proxy cache which is aware of congestion control mechanisms used in transmission of multimedia streams. Streams cached with our mechanism can be used later to perform smoothing so as to facilitate client-side playback buffer management.

There are numerous works on proxy cache replacement algorithms ([1, 11, 15, 18, 24, 23]). They evaluated their algorithms using existing web proxy traces. It's not clear yet how would these algorithms perform when there are a significant number of requests to huge multimedia streams. To our knowledge, there is only one work which addresses the influence of multimedia streams on cache replacement algorithms [20]. They consider resource requirement, e.g., bandwidth and granularity of web objects, during cache replacement. Our work complements it in that we provide more accurate estimation of bandwidth and size requirements through the exposure of congestion control mechanisms. Given this, their algorithms may be easily fit into our architecture. It remains a future work to analyze and evaluate all these replacement algorithms, as well as ours.

# 6 Conclusion and Future Work

This paper discussed implications of the need for congestion control and quality adaptation in the Internet on proxy caching of multimedia playback streams. We justified the necessity for quality adaptation for multimedia streams and identified the limitations of replaying a variable-quality stream from the cache for subsequent requests. We presented our end-to-end architecture for delivery of layered-encoded streams in the Internet and argued that proxy caches complement this architecture. Our layered approach to quality adaptation provides a perfect opportunity to smooth out variations in quality on a demand-driven fashion by pre-fetching of required pieces that are missing in the cache. We also extended the semantics of popularity and introduced the idea of weighted hit to capture both level of client interest and the usefulness of a cached layer. We have described a fine-grain replacement algorithm and illustrated that its interaction with pre-fetching converges the state of the cache to the optimal state where quality of each cached stream is proportional to its quality, and the variations in quality of a cached stream are inversely proportional to its popularity.

We plan to continue our work in several phases. For the first phase, we start with a simulation-based study of the caching mechanism to investigate the impact of different parameters on its performance. More specifically, we are interested in the effect of the following parameters: 1) access patterns, 2) bandwidth distribution between clients, 3) segment size, 4) cached stream length distributions. Furthermore, we plan to study the overhead of conservative pre-fetching as well as other popularity functions. In the second phase, we plan to validate our simulation results by using a real world access pattern from actual traces collected at popular servers in the Internet.

Demand-driven pre-fetching is another interesting area for us to explore where clients specify their interests to a particular stream and its desired quality ahead of time. Providing VCR-functionality by the proxy and the implication of this on caching requires further investigation as well. We also plan to study potential effects of inter-cache architecture on the proxy caching mechanism.

# References


[1] Pei Cao and Sandy Irani. Cost-aware www proxy caching algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 193–206, December 1997.

[2] A. Chankhunthod, P.B. Danzig, C. Neerdaels, M.F. Schwartz, and K.J. Worrell. A hierarchical Internet object cache. In *USENIX Conference Proceedings*, pages 153–63, 1996.

[3] A. Dan and D. Sitaram. A generalized interval caching policy for mixed interactive and long video environments. In *IS&T SPIE Multimedia Computing and Networking Conference*, San Jose, CA, January 1996.

[4] A. Dan and D. Sitaram. Multimedia caching strategies for heterogeneous application and server environments. *Multimedia Tools and Applications*, 4:279–312, 1997.

[5] Asit Dan, Daniel Dias, Rajat Mukherjee, Dinkar Sitaram, and Renu Tewari. Buffering and caching in large scale video servers. In *Proceedings of IEEE COMPCON*, pages 217–224, 1995. http://www.cs.utexas.edu/users/dmcl/papers/ps/COMPCON95.ps.

[6] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *Under submission*, February 1998. http://www-nrg.ee.lbl.gov/floyd/papers.html/end2end-paper.html.

[7] S. Ghandeharizadeh, A. Dashti, and C. Shahabi. A pipelining mechanism to minimize the latency time in hierarchical multimedia storage managers. *Computer Communications*, March 1995.

[8] S. Ghandeharizadeh and C. Shahabi. On multimedia repositories, personal computers, and hierarchical storage systems. In *Proceedings of ACM Multimedia Conference*, 1994. http://perspolis.usc.edu/Users/shkim/papers/94-578.ps.

[9] Shahram Ghandeharizadeh, Roger Zimmermann, Weifeng Shi, Reza Rejaie, Doug Ierardi, and Ta-Wei Li. Mitra: A

scalable continuous media server. *Multimedia Tools and Applications Journal*, 5(1):79–108, jul 1997.

[10] Microsoft Inc. Netshow service, streaming media for business. http://www.microsoft.com/NTServer/Basics/NetShowServices.

[11] S. Irani. Page replacement with multi-size pages and applications to web caching. In *Proceedings of the Annual ACM Symposium on the Theory of Computing*, March 1997.

[12] M. Kamath, K. Ramamritham, and D. Towsley. Continuous media sharing in multimedia database systems. In *Proceedings of the 4th International Conference on Database Systems for Advanced Applications*, April 1995.

[13] Progressive Networks. Http versus realaudio client-server streaming. http://www.realaudio.com/help/content/http-vs-ra.html.

[14] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. *ACM SIGCOMM*, September 1998.

[15] J. Pitkow and M. M. Recker. A simple yet robust caching algorithm based on dynaic access patterns. In *Proceedings of the 2nd International WWW Conference*, pages 1039–1046, October 1994.

[16] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. *To Apprear in Proc. IEEE Infocom*, 1999.

[17] Reza Rejaie, Mark Handley, and Deborah Estrin. Architectural considerations for playback of quality adaptive video over the internet. *Under Submission*, November 1998.

[18] Luigi Rizzo and Lorenzo Vicisano. Replacement policies for a proxy cache. Technical Report RN/98/13, UCL-CS, 1998.

[19] Subhabrata Sen, Jennifer Rexford, and Don Towsley. Proxy prefix caching for multimedia streams. In *Proceedings of the IEEE Infocom*, 1999. To appear.

[20] Renu Tewari, Harrick Vin, Asit Dan, and Dinkar Sitaram. Resource based caching for web servers. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, San Jose, 1998 1998. http://www.cs.utexas.edu/users/tewari/psfiles/mmcn98.ps.

[21] Joe Touch. The LSAM proxy cache - a multicast distributed virtual cache. In *Proceedings of The Third International WWW Caching Workshop*, June 1998.

[22] 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 *Proceedings of the IEEE Infocom*, April 1998. http://www.cs.umn.edu/˜zhzhang/Papers/Wang98:Info98.ps.gz.

[23] Stephen Williams, Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, and Edward A. Fox. Removal policies in network caches for world-wide web documents. In *Proceedings of the ACM SIGCOMM*, pages 293–305, 1996.

[24] R. Wooster and M. Abrams. Proxy caching that estimates page load delays. In *Proceedings of the Sixth International WWW conference*, April 1997. http://proceedings.www6conf.org/HyperNews/get/PAPER250.html.

[25] Lixia Zhang, Scott Michel, Khoi Nguyen, and Adam Rosenstein. Adaptive web caching: Towards a new global caching architecture. In *Proceedings of The Third International WWW Caching Workshop*, June 1998.