# Multimedia Proxy Servers



**Philip Kwok Chung Tse**
*The University of Hong Kong, Hong Kong SAR, China*

## INTRODUCTION

On the Internet, multimedia objects are stored in content servers. The clients behind some proxy servers are located over a wide area network (WAN) far from the content servers (Figure 1). When a client accesses multimedia objects from a content server, the content server must either allocate sufficient disk and network resources to multicast or unicast the objects to the client (Ma & Shin, 2002). Otherwise, it rejects the client. Thus, the popular content server becomes the bottleneck in delivering multimedia objects.

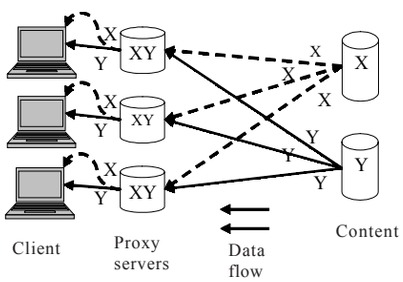
Proxy servers have the disk cache space, network bandwidth, and availability to cache parts of the multimedia objects for clients, making them good candidates to solve the bottleneck problem. However, large multimedia objects are not cached or only partially cached in current proxy servers. When fast optical networks are widely deployed, this problem is becoming more severe. Therefore, proxy caches must be enhanced to alleviate the bottleneck in popular content servers with multimedia objects.

Multimedia proxy servers perform several functions in accessing multimedia objects over the Internet. We first present the background in the next section. Next, the *cache replacement* policies being used in proxy servers are described. Then, the *object partitioning* methods are described. After that, the *transcoding* method that converts high-resolution objects into low-resolution objects is described. Afterward, we present the *cooperative caching* method that can be applied to cache objects on proxy servers. Lastly, we describe a method to distribute proxy-server load using a depot.

## BACKGROUND

One of the main research issues in multimedia object delivery is the provision of *quality-of-service (QoS)* streaming. A multimedia stream is a group of periodic requests that are separated from each other at a fixed time interval. In order to support QoS streaming, the network bandwidth from the content server to each client needs to be able to support the variable data rate of the object stream. Otherwise, some blocks will be missed and jitters will occur. If many blocks are missed, the multimedia stream may even be discontinued.

*Figure 1. Delivery of multimedia contents over the Internet using proxy servers*

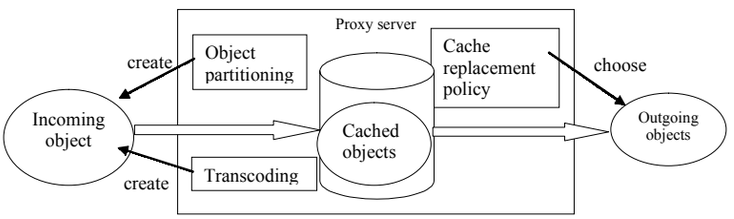



There was much research on enhancing individual proxy servers for multimedia streaming (Chang & Hock, 2000; Ham, Jung, Yang, Lee, & Chung, 1999; Nam & Lee, 1997; Xiang, Zhang, Zhu, & Zhong, 2001). The use of proxy servers, as virtual servers between the servers and the clients, was proposed to manage server-client connections, data delivery, and transcoding (Acharya, Korth, & Poosala, 1999; Chandra, Ellis, & Vahdat, 1999; Nam & Lee). The proxy server is becoming the centre of management to handle server bandwidth limitation and client bandwidth adaptation.

When a proxy server is placed between the content server and its clients, it may store some of the delivered objects in its local cache storage for repeated retrievals. When the cache storage is fully used, it deletes some cached objects to release space for new objects. The cache replacement policy will choose which object should be deleted. The cache content and the cache performance thus depend on the *cost function* of the cache replacement policy (Figure 2). In the literature, many cost functions have been proposed and studied for multimedia objects in a single proxy-cache environment (Acharya et al., 1999; Aggarwal, 1997; Bahn, Koh, Noh, & Min, 2002; Hosseini-Khayat, 1998; Paknikar, Kankanhalli, Ramakrishnan, Srinivasan, & Ngoh, 2000; Sohoni, Min, Xu, & Hu, 2001; Su et al., 2000; Wu, Yu, & Wolf, 2001; Xiang et al., 2001).

When a proxy server caches a large multimedia object, the cache space may not be able to store many objects.

*Figure 2. Relationship among different functions in a single multimedia proxy server*




Instead of storing the entire object, the proxy server may store only a part of it. The cached part can thus be directly retrieved from the local storage and only the missing part is accessed from the content server. The position and size of the cached part affect the network bandwidth requirement and the response time of new streams. In the literature, three object partitioning methods, namely, leader, *staging*, and hot spot, have been proposed for individual proxy servers (Fahmi, Latif, Sedigh-Ali, Ghafoor, Liu, & Hsu, 2001; State & Festor, 2001; Zhang, Wang, Du, & Su, 2000).

When multiple proxy servers are present in a regional network, they may exchange their cache contents to serve client requests more efficiently. The only found research works on the cooperative caching of multimedia objects are on the leader method (Park, Park, & Son, 2001) and the *multiple hot-spot* method (Tse, Leung, So, & Lau, 2003).

## CACHE REPLACEMENT POLICIES

The main contribution or responsibility of proxy servers to the clients is their cache content. The cache content depends on the cost function in the cache replacement policy that determines the cache performance. Hence, the cache replacement policy must be optimized to achieve the lowest capacity miss rates on the cache.

Traditional cache replacement policies considered recency and frequency in the cost function of the cached objects to replace the oldest object in the cache. Xiang et al. (2001) added delays into the cost function. Acharya et al. (1999) added resolution size. Paknikar et al. (2000) added the object size and layer number. Aggarwal (1997) and Wu et al. (2001) increased the segment length when the position of a segment is far from the beginning of a video. This is advantageous when many users stop playing the media after only some initial blocks. Using these cost functions, either the cache hit rate or byte hit rate is optimized for each individual proxy server. In general, Bahn et al. (2002) described the cost function as

$$\text{Cache Value} = \frac{\left(d_i^{r1} * n_i^{r2}\right)}{\left(t_i^{r3} * s_i^{r4}\right)},$$

where $d_i$ is the latency to fetch an object $i$, $n_i$ is the number of references made to $i$ since it has been brought into the cache, $t_i$ is the last reference time, $s_i$ is the size of object $i$, and $r1$, $r2$, $r3$, and $r4$ are constants with default values $r1 = 0.1$ and $r2 = r3 = r4 = 1$.

Web *prefetching* obtains the Web data that a client is expected to need on the basis of data about that client's past surfing activity. It reduces access latency by actively preloading data for clients. Bianchi and Mancuso (2003) minimize the connection outage probability by controlling the buffer size among connections. The prediction by the partial match model (PMM) makes prefetching decisions by reviewing the universal resource locators (URLs) that similar clients have visited (Chen & Zhang, 2003). The model forms a Markov predictor tree structure of these URLs. The standard PMM builds a tree for every visited URL. A fixed threshold is used to limit the length of each prediction branch. In order to achieve accurate prediction on future requests, the tree being built is very large and consumes too much space.

The longest repeating-sequence (LRS) PMM reduces the size of the prediction tree by storing only long branches with frequently accessed URL predictors. The tree size is reduced at the expense of lowered prediction accuracy.

The popularity-based (PB) PMM ranks a URL's relative popularity (RP) into four grades:

- Grade 3: 10% < RP <= 100%
- Grade 2: 1% < RP <= 10%
- Grade 1: 0.1% < RP <= 1%
- Grade 0: RP <= 0.1%

It assigns long branches to popular URLs and shorter branches to less popular URLs (Chen & Zhang, 2003). Thus, only frequently accessed paths are kept to reduce the storage requirement of the predictor tree.