Chapter 2

Background

2.1 Introduction

In the past few years, the client requests for streaming video over the Internet have become much popular and this trend is expected to continue. Therefore, caching multimedia objects have become increasingly vital to meet the requirements of users and at the same time handling the huge Internet traffic and bandwidth consumption. Hence, most of the research is focused on the caching media on the proxy server in different aspects. In this chapter, literature review of the proxy caching for multimedia streaming is discussed in detail.

On the Internet, the proxy servers are placed near to the clients. Typically the delay between the proxy and the client seems negligible, but it is in fact quite large between the proxy and the content provider. As stated in section 1.2.3, the user requests the media through the proxy and if these objects have already been cached and valid, they are returned immediately. If not, the proxy will fetch these requests from the content provider or even other proxies and send them back to the clients, as well as cache these objects locally. Also, in order for a client device to instantly play back the media, there must be enough initial blocks at the proxy server to mask the latency between it and the content server.

Streaming media data have different characteristics from traditional web data (text and image); hence, existing caching techniques are not appropriate for it. It would be impractical to cache the whole content of a movie due to the size of the video objects. That would consume disk usage in the proxy so large that there could be only a few movies on a whole disk. Also, due to user demand, the number of video is becoming higher and higher as more network bandwidth becomes available.
Because of limited disk space and bigger files, streaming media can be better cached on disk either whole or part to satisfy the storage consumption (one of the important parameter) of an object in the proxy. Of course, there are the trade-offs in deciding the ratio of caching of objects. Another important factor to satisfy customer desire is delay sensitivity. By nature, unlike traditional traffic, a video stream should be real time and continuous. In proxy caching research, it is quite difficult to trace the exact client behavior from network traffic in order to achieve the optimum caching strategy; for example what should be cached if the clients query only a few portions of movie files, or how to implement caching strategies to support VCR-functionality in the case of a finite cache [17]. Hence, a very tedious task is finding optimum caching methodology to decide which parts of the media object to be cached for the newly loaded objects from the origin server to satisfy the above said parameters.

To understand the existing proxy caching methods, extensive study has been made on scalable and non-scalable caching methods discussed in section 2.2 and section 2.3 respectively. Another category of proxy caching method is cooperative proxy caching, which is combined with any one of the above caching methods. Various cooperative caching methods are discussed in section 2.4. The additional supports in proxy caching methods like random seek and cache replacement policies are also viewed and presented in section 2.5 and section 2.6 respectively.

2.2 Scalable Proxy caching methods

In the scalable proxy caching model, multimedia objects are QoS adaptive caching [18], [28], [34], [36], [104] and coded either using scalable coding by layered coding techniques [19]–[27], [29]-[31], [59] or the proxy server is transcoding-enabled [32]-[33], [74], [105].

2.2.1 QoS adaptive Proxy caching

In QoS adaptive caching, multiple versions of an object are stored in the origin server. The proxy server can cache the all versions or few versions of objects from the origin server depend on the requirement of the heterogeneous clients.
Reza Rejaie et al developed Mocha [18], a quality adaptive multimedia proxy cache for Internet streaming on top of Squid. The mocha performs quality adaptive caching. Mocha caches popular streams and adaptively adjusts the quality of cached streams based on the stream popularity and available bandwidth to the interested clients. The mocha is maximizing delivered quality to heterogeneous clients without compromising cache space utilization.

Fang Yu [28] et al proposed a QoS-adaptive caching scheme for mixed media. The scheme divides the physical cache into several blocks. Each block store a specified type of media and the page size in the block is equal. However, different blocks have different page sizes. The page size for each type of media is determined based on the statistics result obtained from the typical traces. The space each type of media occupies is variable and adaptively changes according to the hit/miss ratio. After the specific period of time, the scheme enlarges the space of the media with high hit-ratio and decrease the one with high miss-ratio.

Bing Wang[34] et al presented optimal proxy prefix cache allocation to address the problem of efficiently streaming a set of heterogeneous videos from a remote server through a proxy to multiple asynchronous clients so that they can experience playback with low startup delays. In this model, proxy caching integrated with traditional server-based reactive transmission schemes such as batching, patching and stream merging to develop a set of proxy-assisted delivery schemes. The scheme also provided a set of reactive transmission schemes that use proxy prefix caching as an integral part for bandwidth-efficient delivery in an Internet-like environments, where the end-to-end network connections only provide unicast service, or at best offer multicast capability on the proxy-client path.

Kaihui Li [36] developed an Optimal Prefix Caching and Data Sharing (OPC-DS) model for scalable multimedia streaming. The scheme is a combined model of prefix caching and interval caching and implemented at the proxy cache. In OPC-DS, the sizes of the appropriate prefix cache and interval cache are calculated according to the current request distribution. In OPC-DS, a portion of proxy cache space is used to
cache the video prefixes. The remaining cache space is used to dynamically cache the uncached portion of the videos as interval cache.

2.2.2 Layered encoding proxy caching

In a layered encoding scheme, the video information is encoded into several layers and it is applied in heterogeneous client environments [24]-[27], [29]. The base layer carries important video and critical timing information and the higher layer improves the quality of the video gradually. The receiver can get a reasonable quality with the base layer, and the quality improves with the reception of higher layers. The encoder allocates priorities to the encoded layers with the base layer having the highest priority. When the network transmits layered video, it can drop low priority (higher) layers in the event of congestion.

The proxy caching scheme for the layer encoded video, the following factors are considered into account (i) it is no need to cache one whole clip layer at each proxy entirely, partial segment caching can provide better caching gain. (ii) Segments layout must answer for segment popularity in addition to clip popularity. (iii) At last, in the heterogeneity of access, the Internet should be taken into account when calculating the probability of segments.

Bhofeng Liu [20] et al developed a distributed layered encoding video based on popularity of segments. The popularity is derived from the popularity of inter-video clip and intra-video clip as well as distribution of users' downstream bandwidth. This method mainly addressed and analyzed about how to divide segments and decided which segment should be stored in the limited capacity cache in order to minimize the distant server hit ratios.

Niu Xianlong [22] et al presented a cache scheduling scheme based on layered coding VoD system. A local cache scheduling scheme based on layered coding is introduced to improve QoS of Internet VoD service. The base layer data units and enhancement layer data units are defined as BB (base layer base unit) and EB (enhancement layer base unit). Data is organized as base layer units and enhancement
layer units in the local cache, which are used as units for caching and replacing.

Dimitris N. Kanellopoulos [23] presented Adaptive Multimedia Streaming to Changing Network Conditions. Various techniques are discussed such as layered encoding, receiver driven multicast, rate shaping, error control viz forward error correction (FEC) techniques for Internet audio and video, adaptive synchronization, and smoothing for achieving adaptation at the application layer.

Reza Rejaie [59] et al proposed layered-encoded multimedia streams at Internet proxy servers, and included a prefetching scheme to smoothen out the variations in quality of a cached stream during subsequent playbacks. The scheme extended the semantics of popularity and introduced the idea of weighted hit to capture both levels of interest and usefulness of a layer for a cached stream. The weighted hit ratio of a cached stream is used as a metric to measure its popularity. The weighed hit is calculated by the ratio between the playback time and stream length where playback time and stream length denoted total playback time and length of the entire stream respectively. The proxy server keeps track of weighted hits on a per-layer basis. For each layer of a session, the total playback time is recorded and used to calculate the weighted hit for that layer at the end of the session.

**2.2.3 Transcoding enabled proxy caching**

Transcoding is the direct analog-to-analog or digital-to-digital conversion of one encoding to another such as for movie data files or audio files. This is usually done in cases where a target device (or workflow) does not support the format or has limited storage capacity that mandates a reduced file size, or to convert incompatible or obsolete data to a better-supported or modern format. Transcoding is commonly a lossy process, introducing generation loss; however, transcoding can be lossless if the input is losslessly compressed and the output is either losslessly compressed or uncompressed.

Users in fast networks prefer high resolution videos while users without high-speed network access may not enjoy high quality videos because the delay is large and
the video may not fit within the device display. Many content providers encode streaming video clips at several different bitrates (28 to 56 kbps for dial-up connections and 150-plus kbps for broadband networks) to satisfy users with different network connection speeds. Various bit rate versions of the same video clip may be cached on the proxy at the same time, which is a waste of storage.

The Transcoding-enabled Caching (TeC) proxies for multimedia streaming distribution over the Internet are suited for heterogeneous network environments and client capabilities. Depending on the connection speed and processing capability of end user, the proxy transcodes the requested (and possibly cached) video into an appropriate format and delivers it to the user. By putting the transcoding unit in the content delivery path, content adaptation is performed at the network edges (usually at proxy servers). The potential advantage of TeC is that the origin servers need not generate different bit-rate versions (although multiple versions are available) but done on the proxy servers.

Video transcoding is a computation-intensive task. Many transcoding enabled proxy caching schemes have developed to reduce the workload of a transcoding session [74]. Among those, compressed domain based approach provides the best performance. In compressed domain transcoding, the input video is only partially decompressed. Rate adaptation is performed in the compressed domain while the motion information is reused. This approach considerably improves the processing speed over the conventional decode-and-re-encode approach. The TeC proxy utilizes compressed domain transcoding techniques.

Bo Shen [32] et al developed caching strategies in transcoding-enabled proxy systems for streaming media distribution networks. The scheme has three caching algorithms for TeC which take into account that variant of the same video. The first two algorithms cache at the most one version of a video object. They operate differently when a user requests a video version that is coded at a lower bit-rate than the one cached in the proxy. The third algorithm may cache multiple versions of the same video object to reduce the processing load at the transcoder.
Yoohyun Park [33] et al proposed a hybrid segment-based transcoding proxy caching of multimedia streams. In this proposal, it is assumed that an object has multiple versions at variable bitrates. Hybrid segmentation is a uniform segmentation in the same version, but it is exponential segmentation among different versions. Consequently, the total number of segments in each version is the same. So, if all versions of the same content are played at the same time, the consuming number of segments at the given time is the same.

2.3 Nonscalable proxy-caching schemes

The nonscalable proxy-caching schemes support and deliver the single layered CBR video objects to the clients. It is categorized into two classes such as the prefix/suffix approaches [37]-[41], [56], [65] and the segment-based [42]-[56] approaches. The prefix/suffix and segment based caching also supports in the scalable environments to support the layered and transcoding enabled proxy caching schemes. But these schemes are primarily used in the non-scalable environments. To provide additional fairness on the proxy caching, the popularity caching schemes [39], [53], [54], [60]-[64] are evolved with the popularity of the video segments.

2.3.1 Prefix/Suffix Caching

The prefix/suffix caching schemes divide a streaming media object into two atomic caching blocks, prefix and suffix. Prefix caching algorithm caches the initial portion of a media object, called the prefix, at a proxy. Upon receiving a client request, the proxy immediately delivered the prefix to the client, while fetching the remaining portion, the suffix, from the origin server and relaying to the client. The structure of the prefix/suffix is shown in Figure 2.1. Many prefix/suffix schemes [34]-[41], [51], [56], [65] are developed for multimedia streaming, over a span of years and few among them are discussed below.
Chapter 2

Figure 2.1 Prefix/Suffix parts of a video segment

Dongliang Guan [35] developed optimal prefix cache allocation for multimedia streaming. In this scheme, the optimal prefix length of each video to be cached on a proxy is calculated approximately in direct proportion to the square root of the video’s total length.

Li Zhu [37] et al presented a Proxy Caching for Video on Demand System for Multicasting Networks which used the prefix caching model. The system model contained one central server and a total number of $P$ proxies. Each proxy has stored the beginning of each video file (prefix), and transmits the prefix to their serviced clients. The server could stream the remaining video files (suffix) directly to clients instead of through the proxies. The server, as well as proxies, used the bandwidth skimming scheme to deliver video streams and each client can receive data from maximum two streams at the same time.

Dakshayini M [38] et al proposed an efficient prefix caching scheme with unicast communication. The proxy server is termed as the parent server (PS) and client received the prefix part of the PS. Based on the frequency of user requests to any video, the popularity of the videos and size of prefix to be cached at PS are determined. This arrangement allows the PS to cache maximum portions of a number of most frequently requesting videos.

GopalaKrishnan Nair R [39] et al proposed an efficient regional popularity based prefix caching and load sharing algorithm (RPPCL). The algorithm efficiently allocated the cache blocks to the video according to their local popularity and also shared the videos present among the Proxy Servers (PS) of the Local Proxy Servers Group (LPSG). The proposed model is aimed to increase the video hit rate and it
reduces the client’s waiting time, network usage on Main Multimedia Server (MMS) to PS path.

Dakshayini M [40] et al proposed an optimal regional popularity based video prefix replication strategy and a Scene Change (SC) based replica caching algorithm that utilizes the zipf-like video popularity distribution to maximize the availability of videos closer to the client and request-servicing rate thereby reducing the client rejection ratio and the response time for the client. A scene change is identified if the change in the number of bits between successive frames exceeds a certain threshold in a continuous manner.

Yuan-Tse Yu [41] et al presented an adaptive suffix-window (ASW) caching scheme for continuous media m proxy server. In suffix-window caching mechanism, the proxy caches a suffix window which is dynamically scheduled. The suffix window is comprised of some playback segments that have been run recently. In the ASW scheme, the proxy always caches the prefix port and caches a suffix-window long segments that have been run recently.

Wei Tu [51], Lian Shen [56] proposed a prefix segment mechanism for a flexible starting point for video on demand. The scheme splits a video first into segments with a length of ‘\(M\)’ Group of Pictures (GOPs). The first ‘\(N\)’ GOPs in a segment are defined as prefix and the remaining GOPs in a segment are defined as a suffix. The size of the prefix ‘\(N\)’ is adjusted to the round trip time and the transmission capacity between the remote server and the proxy. The size of the prefix is always less than or equal to ‘\(M\)’.

Alan T.S. Ip [65] et al developed a prefix/suffix caching algorithm with the client supports in caching. The proxies are responsible to cache the prefix of video, whereas the clients cache the prefix-of-suffix of video. When a client expects to play a video, it first initiates a playback request to its home proxy, which intercepts the request and computes a streaming schedule: when and where to fetch which portion of the video. It then accordingly fetches the prefix, prefix-of-suffix, as well as the remaining part of the suffix, and relays the incoming stream to the client.
2.3.2 Segment proxy caching

In segment based proxy caching, the video file is divided into multiple equal-sized blocks. Multiple blocks are then grouped into various segments by the proxy server. In fixed/uniform segmentation, each segment contains a fixed number of blocks. With a smaller segment size, the cache byte-hit ratio is generally higher. The number of segments in the cache can be large, so the cache management overhead increases. On the other hand, with a larger segment size, cache management is more efficient, but it is achieved at the expense of a lower cache hit ratio. In uniform segmentation, objects are segmented according to a uniform length as shown in Figure 2.2.

![Figure 2.2 Fixed segment structure with M prefix GOPs and N suffix](image)

In exponential segmentation, objects are segmented in a way that the size of a succeeding segment can double the size of its preceding one. These segmentation-based strategies favor the caching of beginning segments of media objects. The third method of segmentation is dynamic segment caching [51] based on either the popularity of the video or user access pattern of the video. It contains varying number of segment sizes and varying number of prefixes cached.

![Figure 2.3 Dynamic Segment Caching with varying prefix and suffix](image)
Xiaoling Li [42] et al proposed a dynamic segment-based caching algorithm (DECA) for video on demand based on the popularity of the video. It adjusts the segment size according to its hitting rate and maximizes the user satisfaction by optimally trading off between the initial delay and the deviation of the starting point. In the DECA, the prefix length of the segment is determined by the network condition between the proxy and the server. However, the suffix size is dynamically adjusted according to the popularity and current length of the segment. The popular parts with small suffix size or even no suffix, are assigned, that is, the length of the segment equals to the length of the prefix. By dynamically adjusting the segment size, it is ensured that the prefix frames of all segments are cached on the proxy, which led to a possible “no waiting time”.

Kun-Lung Wu [43] et al studied three media segmentation approaches to proxy caching: fixed, pyramid, and skyscraper. The first approach is called fixed segmentation, where each segment size is fixed. In this approach, the cache management overhead can be large, especially when the segment size is small and the total number of segments is large. The second approach is called pyramid segmentation, where segment size increases exponentially from the beginning segment. The third approach is called skyscraper segmentation. It is a variation of the pyramid segmentation, but the segment size increases slowly. For example, the size of segment $i+1$ is either the same as segment $i$ or twice as that of segment $i$. When stacked together vertically, the segments look like a skyscraper instead of a pyramid. Skyscraper segmentation has more segments to manage than pyramid segmentation have, but have lesser segments than fixed segmentation.

Songqing Chen [45] et al developed a segment-based proxy caching called as Hyper Proxy addresses the issues in this model like improving the byte hit ratio and reducing proxy jitter, and reducing the delayed startup ratio. To meet the above two factors, the scheme adopted an active prefetching method along with segment caching.

Songqing Chen [46] et al designed and implemented a segment-based streaming media proxy, called SProxy. This proxy system produced three enhancements in the
proxy caching. Firstly, *SProxy* leveraged the existing Internet infrastructure to address the flash crowd. The origin server is now free of the streaming duty while hosting streaming content through a regular Web server. Thus, UDP based streaming traffic from *SProxy* suffers less dropping and no blocking. Secondly, *SProxy* streams and caches media objects in small segments determined by the object popularity, causing very low startup latency, and significantly reducing network traffic.

Muhammad Muhammad [47] et al presented an analytical model for the evaluation of the performance of a VoD system and estimated the mean waiting time achieved by the Popularity-Aware Partial caching (PAPA) algorithm [56]. The scheme used two segment-prefix structure methods such as popularity first approach and fairness first approach. The waiting time is further minimized, when dropping more data from unpopular contents with the popularity first approach. It performed better than normal PAPA that works at the frame level. The fairness first approach performed the worst as prefix frames are evenly dropped without taking into account their individual popularities.

Putra Sumari [48] et al proposed a caching scheme which denoted the Proxy Server Cache Mechanism (PSCM) to overcome the start-up service delay. In this scheme, the first segment of all broadcasting videos is stored on a stationary proxy server. The late clients can request the first segment directly from proxy server instead of waiting for the next broadcasting channel. Balancing technique is used to maintain the clients’ requests on the local proxy and ensure a fair distribution of these requests.

James Z. Wang [49] developed a Fragmental Proxy Caching (FPC) for streaming multimedia objects. The novel data-fragmentation method not only provided finer granularity caching units to allow more effective cache replacement, but also offered a unique and natural way of handling the interactive VCR functions in the proxy-caching environment. The video file is divided into tiny fragments (smaller blocks) and classified as *Cfrag* and *Sfrag*. The *Cfrags* units are cached and *Sfrags* are streamed from the origin server while playing *Cfrags*.
Lei Guo [50] et al developed a Dynamic Interleaved Segment Caching for Interactive Streaming (DISC). The DISC trades cache performance for response time to client interactive requests. In DISC, segments of a media object are cached dynamically according to client access patterns. DISC supported direct jumps efficiently while ensuring timely prefetching of uncached segments for sequential accesses. In DISC, each object cached fully when it is accessed for the first time to set up an observation period for the client access pattern. Later, when the fully cached object is selected as a victim by the replacement policy, a quota is calculated to determine the number of segments that could remain in the cache. By caching objects in interleaved segments, the DISC sacrificed proxy cache performance in order to reduce the response time to client jump requests.

Wei Tu [51] et al proposed an efficient dynamic segment-based caching algorithm, which maximizes the user satisfaction by trading off between the initial delay and the deviation of the starting point. The caching algorithm applies prefix/suffix caching method at the initial time of object caching. Later, it dynamically adjusts the caching size of a segment based on the popularity earned for the object.

Songqing Chen [44], [52] et al proposed an adaptive and lazy segmentation based proxy caching for streaming media delivery. This adaptive and lazy segmentation mechanism delayed the segmentation as late as possible and determined the segment length based on the client access behaviors in real time. In addition, the admission and eviction of the segments are carried out adaptively based on an accurate utility function. In this model, an aggressive admission policy is used in which, the object is fully admitted when it is accessed for the first time. Then the admission of this object is considered segment by segment based on the user’s access logs. The lazy segmentation strategy adaptively set different base segment lengths for different objects according to real time user access behaviors.
2.3.3 Popularity based proxy caching

Popularity of media object and most watched portion of an object may vary from time to time. For instance, some objects may be popular only for an initial time period when most users access the entire object. In this scenario, using a fixed strategy of caching first few segments may not work. The reasons are as follows: When the object is popular and not cached properly, it may overload the network because the system needs to retrieve the remaining segments for each client access. On the other hand, as the object’s popularity diminishes, caching the initial segments may waste proxy resources. The lack of adaptability in the existing proxy-caching schemes might render proxy caching ineffective. Hence, many proxy caching schemes [18], [39], [53], [54], [61]-[64] are developed based on the popularity of the segments either considering internal popularity or object popularity.

The amount of content to be cached can vary dynamically according to popularity variation of each content. That is, if the amount of requests by clients increases, the proxy server caches more segments for that content. On the contrary, if it decreases, the proxy server caches less data. The contents that have very high popularities can be cached entirely, but the contents that have low popularities can be cached of very small amount of prefix or evicted from cache, entirely. In this way, the contents have occupied proper cache space according to their popularities.

Jiang Yu [54], [61] proposed a new caching algorithm called as Internal Popularity Based (IPB) caching algorithm. This algorithm estimated the internal popularity of each video based on k-transformed Zipf-like model [102], [103] and chosen the appropriate segments to cache, to minimize the bandwidth consumption of the network.

Beomgu Kang [63] et al investigated a combination of popularity and segment based caching management scheme. In this scheme, after dividing equal sized segments of a video, each segment is further divided into equal-sized sub segments. The popularity of the video object is calculated based on maintaining the hitting table. In the hitting table, a hit count variable is maintained for each object and whenever a
user requests an object, the hit count is increased. The objective of this scheme is to maintain the cache space of each stream to be proportional to its popularity.

Most accesses are for a few popular objects, and it’s likely that clients will watch these in their entirety or near entirety. This is often true for movie content in a video-on-demand (VoD) environment and training videos in a corporate environment. However, a proxy is always exposed to objects with a wide range of sizes from different categories, and the access characteristics of these objects can be quite diverse. Without an adaptive scheme, an overestimate of the base segment length may cause an inefficient use of cache space, while an underestimate may induce increased management overhead.

As a summary, the non-scalable proxy caching methods are very easy to implement at proxy server due to the nature of single layered CBR videos. The proxy jitter is very low and complexity of caching process is very less. It is also easy to main the popularity of segments in the proxy server and in the origin server.

### 2.4 Cooperative proxy caching

The cooperative proxy caching schemes [38], [64]-[73] are introduced to utilize the cache space effectively, reduce the bandwidth consumption and latency. The media objects are segmented into equal-sized segments and stored across multiple proxies, where they can be replaced at a granularity of a segment. There are several local proxies called as home proxies responsible to answer client requests by locating and relaying the segments.

The cooperative proxy caching schemes are used to cache the segments either in cooperated proxies or both client and proxies and illustrated in Figure 2.4. The major challenges in these schemes are to allocate appropriate cache space in the proxies and to find the corresponding proxy which cached the segments. The client storage space is also used to cache in addition to proxy caching system. These schemes increased the additional responsibility to the clients to extend the cache allocation and lookup process at the client level and proxy level.
The overhead for the proxy cooperation system is determined by three separate cache functions such as discovery, dissemination, and delivery. Discovery refers to how a proxy locates cached objects. Dissemination is the process of selecting which objects are to be cached and transferring those objects from the origin servers to the caches. Delivery defines how objects are delivered from the caching system to the client at the time of the request.

Zeng Zeng [64] et al studied the server-side cooperative proxy system for large scale multimedia applications. It is composed of one multimedia server and $N$ caching proxies. The proxies are connected with the multimedia server directly through an intranet or the Internet, in order to share the requests arriving at the system. The proposed ART-Greedy algorithm supports both static and dynamic situations.

Alan T.S. Ip [65] et al developed a Cooperative Proxy Caching (COPACC) model with prefix/suffix caching to support both heterogeneous and homogeneous networks. In this model, the proxy caching is done at two levels such as client level and proxy level. It leveraged the client-side caching to amplify the aggregated cache space and relies on dedicated proxies to effectively coordinate the communications. It also had an efficient cache allocation algorithm for distributing video segments among the proxies and the clients.
M. Dakshayini [71] et al presented cooperative proxy servers with caching to provide high QoS video streaming. The proposed architecture consists of a Centralized Multimedia Server (CMS), which is connected to a group of trackers (TRs) where each TR is in turn connected to a set of Proxy Servers (PSs). These PSs are connected among themselves in a ring fashion. And also to each of these PSs, a large number of users are connected Local Proxy Servers Group (LPSG). All these LPSGs are interconnected through their TR in a ring pattern. The TR caches some portion of video according to the frequency of requests for video at any one PS only once. Then PS transmits cached data to the client through LAN using its less expensive bandwidth. The TR is also a PS with high computation power, which coordinates and maintains a database that contains the information of the videos present in each PS and also free cache space at each PS.

Cehn-Lung Chan [72] et al presented a cooperative proxy scheme, consisting of a streaming cache and a query protocol revised from Internet Cache Protocol (ICP), to improve performance of large-scale VoD systems. Yoshiaki Taniguchi [73] et al proposed quality-aware cooperative proxy caching for video streaming proxies which communicate with each other and retrieve missing video data from an appropriate server by taking into account transfer delay and offerable quality. In addition, the quality of cached video data is adapted appropriately at a proxy to cope with the client-to-client heterogeneity in terms of the available bandwidth, end-system performance, and user preferences on the perceived video quality.

These schemes proved that well-recognized proxies grouped together can achieve better performance than independent standalone proxies. Cooperating proxies must trust that the objects they accept from other caches have not been modified, and that cooperating partners will not act in a manner contrary to the common good. But cooperative caching that is not administered by a central authority would be difficult to implement. In that case if the central administrative machine failed, the cooperative proxy caching model will also fail. So, the implementation of cooperative proxy caching is very complex.
2.5 Random Seek Supported Caching

A media player maintains a running buffer, which is preloaded with media data before playback starts, to smooth possible playback jitter due to the fluctuation of end-to-end bandwidth between the server and the client. The buffer size ranges from 5 seconds to 30 seconds media data for most of the commercial media players. In streaming multimedia applications, some users want to perform VCR functions, such as fast forward, rewind, and random access, before finishing the playback of the entire object.

Old media data are drained out and new media data is filled into the buffer continuously during streaming. When a user clicks the play button or drags the playback progress cursor on the media player to start a streaming session or to specify a seeking position, the user usually has to wait for a period of time before the playback begins. This waiting time mainly consists of three parts: the round trip time for communication between the client and the server, the server response time from the time when the server receives the command to the time when it sends the first demanded data packet, and the buffering time to fill the media player buffer. Distributing the cached parts across an object allows multimedia streaming to perform in either forward or backward direction. This is an essential feature for handling interactive VCR functions in proxy server environments.

The Continuous Segment Caching (CSC) strategies cached the segments sequentially. Such a design can reduce startup latency, provide free-of-jitter delivery, and improve cache performance based on object popularity. However, they lack the support to jump requests like once a client jumps to an uncached segment, the client will experience a long delay due to the buffering time. The proxy caching schemes [42], [49]-[52], [56], [58]-[59] are developed to focus on random seek while watching the videos to support like VCR functions.

Xiaoling Li [42] introduced the random seek support in VoD. In most of the VoD systems, random access of the video is typically performed by some kind of slide bar, which is not very accurate. However, this inaccuracy seems quite tolerable for the
viewers compared with the perceived waiting time by each request. Based on this observation, a fixed segment-prefix caching structure is proposed. An obvious conclusion drawn is that if the initial delay is not smaller enough than the early start time, viewers would prefer immediate playout of the video with a small offset from the desired starting point.

The fragmental proxy-caching scheme [49] handles user interactive VCR functions due to its symmetrical object fragmentation. In this scheme, only a fraction of the processing unit has been downloaded after fast forward or rewind, no matter where the VCR functions ending. In addition, the media server needs only to deliver the uncached fragments to the proxy server during the resumed regular playback, since the cached fragments are already in the proxy server.

Lei Guo [50] combined the user interactive requests with the proxy caching. It supports jump accesses by caching appropriate segments discretely according to the clients’ jump patterns. In the meantime, it ensures the continuous streaming delivery for sequential accesses with prefetching support whenever necessary. The scheme analyzed that the user behavior on random seek and showed that the jump access (48%) and pauses (51%) are the dominant client interactive requests and that jump access often suffer serious delays due to slow buffering through the network.

Wei Tu [51] proposed a caching scheme supporting interactive VCR functionalities. In this scheme, a user satisfaction score is introduced and based on the score, the proxy caching and replacement are carried out. Lian Shen [56] proposed a popularity-aware partial caching algorithm for VoD which considered the random access of user behavior. The users are able to seek the video and start the play out with virtually no initial waiting time for popular video segments, although only a part of the video is stored on the proxy.

Yuan He [58] proposed a VCR-oriented VOD for large-scale networks. By mining associations inside each video, the segments requested in VCR interactivities are predicted based on the information collected through conversations. For a particular VCR control, the response latency is defined as the duration of interruption
before the playback is resumed. The scheme adopted prefetching methodology to support VCR like interactivities.

Reza Rejaie [59] proposed a proxy caching model that perfectly suited 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. Intuitively, the popularity of each stream must reflect the level of interest that is observed through this interaction. It is assumed that the total duration of playback for each stream indicates the level of interest in that stream.

In the context of streaming applications, the client can interact with the server and perform VCR-functionalities (i.e. Stop, Forward, Rewind, Resume the session). Hence, it is very essential to provide VCR like interactions in the video streaming in addition to the consideration of byte hit ratio, reduction of bandwidth consumption, optimizing the proxy storage space in the with proxy caching. From the study, we understood that the larger size segment division does not provide fairness on VCR functionalities. But the smaller size segment division provides easy to support of VCR functions, because all smaller segments will have some cached part.

### 2.6 Cache replacement policies

The continuous caching of video object using any one caching methods described in the above sections will exhaust the cache space in the proxy server. Hence, the proxy cache space must be made free to accommodate the new objects by removing the existing cached objects either partially or fully. The optimization has been done with various cache replacement polices [26], [30], [42], [45], [48]-[52], [57], [59], [65] [79]-[89], [91]-[101]. Most of them selects an entire cached object or part of the cached object as a victim and apply the policy on it. The victim objects are identified based on the cache requirement and the popularity of the existing objects. There are two types of cache replacement policies such as single-factor replacement policy and multifactor replacement policy. Single-factor replacement policies include LFU, LRU, FIFO, and RAND [30], [42], [59], [82]-[83], [87], [91]-[97], [101]. Multifactor replacement policies comprise of many factors, such as access frequency, recency, latency, and size, which are called weighted functions [26], [45], [48]-[52],
A different classification about cache replacement policies is given in Jin [92] et al as follows:

- **Recency-based strategies** incorporate recency (and size and/or cost) into the replacement process. These strategies use recency as a main factor. Most of them are more or less extensions of the well-known Least Recently Used (LRU) strategy. LRU has been applied successfully in many different areas. LRU is based on the locality of reference seen in request streams. Simple LRU variants do not combine recency and size in a useful, balanced way. They do not consider frequency information. This could be an important indicator in more VoD environments.

- **Frequency-based strategies** incorporate frequency (and size and/or cost) into the replacement process. They are based on the fact that different Web objects have different popularity values and that this popularity values result in different frequency values. Frequency-based strategies track these values and use them for future decisions. Least-Frequently used (LFU)-based strategies require a more complex cache management. LFU can be implemented, for example with a priority queue. Many objects could have the same frequency count. In this case, a tie breaker factor is needed.

- **Recency/frequency-based strategies** consider both recency and frequency under fixed or variable cost/size assumptions. Due to special procedures, most of these strategies introduce additional complexity.

- **Function-Based Strategies** use a potentially general function to calculate the value of an object. They do not assume a fixed combination of factors or fixed usage of data structures. There exists no built-in bias for some objects. Through the proper choice of weighting parameters, one can try to optimize any performance metric. They consider a number of factors for handling different workload situations.
Tz-Heng Hsu [26] et al proposed function based cache replacement for video streaming. The replacement decisions are taken at the block level rather than the entire objects. In this cache replacement policy, the transcoding delay and popularity of videos are considered to calculate the profit of caching ($\text{CacheProfit}$) a video object in the proxy. The proxy computed the $\text{CacheProfit}$ value of the requested segment $v$ of the video. A segment $u$ of other videos is removed from the cache space if its $\text{CacheProfit}$ is less than that of the requested segment $v$. While the proxy had enough space, the requested segment $v$ is stored in the cache space. If the $\text{CacheProfit}$ value of the requested segment $v$ is smaller than that of other videos, segment $v$ is not stored in the cache.

Anna Satsiou [30] proposed a LRLFU scheme combined both LRU and LFU cache replacement management policies. The LRLFU is preferable when there are small and regular popularity changes, that is, a progressive change in the popularities of videos, or large but less frequent popularity changes. The LRLFU cache replacement scheme is well suited for both homogeneous and heterogeneous environments.

Xiaoling Li [42] proposed a dynamic segment based cache replacement algorithm. In this method, a cost of the object is determined based on the two factors such as popularity of the object and user satisfaction. The algorithm works in two steps based on the “cost” metric. It first browses through all videos, checked the segments in the same video pair wise, calculates the cost of each pair and saved it as side information. After all the side information available, the algorithm sorted the cost values of each segment pair in ascending order and started merging from the first pair, i.e., the one with the smallest cost. Merging a segment pair means to set the whole second segment as part of the suffix of the first segment. All GOPs of the second segment in the pair are completely removed so that free space is created in this process.

James Z. Wang [49] proposed a novel cache-replacement scheme that uses a sliding history window to monitor the dynamic user request arrivals and employs a
tunable victimization procedure to provide an excellent method of managing the cached multimedia data according to different QoS requirements of streaming multimedia applications. The cache-replacement algorithm is designed by considering both the user request arrival rates for individual multimedia objects and the playback rates of these objects in selecting replacement victims. These two factors are used to calculate the average playback rate (APR). The cache replacement is done by selecting the objects with the least APR values.

Lei Guo [50] proposed popularity based cache replacement policy for video streaming. The replacement policy looked for the fully cached object first. Among the fully cached objects, the oldest one is always selected as a victim. If no sufficient space is found after all the fully cached objects are checked, the popularity based replacement policy applied to partially cached objects. The replacement policy always replaced the segments of the least popular object, based on popularity of the object from its tail until sufficient cache space is found. The replacement scheme also introduced a self-replacement algorithm which compares the popularity of cached segments to uncached segments in the same chain to determine if the cached segments are sufficiently popular. If not, the cached segments are replaced with more popular uncached segments if the popularity of the cached chain is less than any of the uncached ones by the threshold for popularity ratio changes. The popularity measure could be either the number of jump accesses or the number of total accesses.

Wei Tu [51] et al proposed a popularity based cache replacement algorithm. It keeps most of the segment pair unchanged and only worked on the fragments which had large changes in their popularity. For those having a decreased popularity, segment pairs will be merged to leave some space. Meanwhile, the segments with increasing popularity split into two small segments (e.g., evenly), recursively. The replacement algorithm always takes an initial segment structure as its input.

Songqing Chen [52] et al proposed a two phase replacement policy to keep more objects in the cache with the help of utility function. The utility function is derived to help the victim selection process using several factors such as the average
number of accesses, the average duration of accesses, the length of the cached data and the probability of the future access. In the first phase, if the access log of the object indicates that the object is fully cached, the object is segmented using lazy segmentation. The first two segments are kept and the rest of the segments are evicted right after the segmentation is completed. In the second phase, if the access log of the object indicates that the object is partially cached, the last cached segment of this object is evicted.

Chi-Feng Kao [79] et al proposed Aggregate Profit-Based Caching Replacement Algorithms for Streaming Media Transcoding Proxy Systems. This proposed function takes into account the popularity of certain versions of an object, the transcoding delay among versions, and the average duration of access of each version. Based on the video profit function, the cache-replacement algorithm is used to reduce the startup delay and network traffic by efficiently caching video objects with most profits. The complete video profit function is formulated for estimating the profit from caching a version of a video object when other versions of the object are already cached. ElAarag [80] et al. compared function-based proxy cache replacement schemes based on the hit rate, byte hit rate, and removal rate. They suggested that the LRU and GDSF methods are not suitable for video streaming, whereas function-based methods such as M-Metric, MIX, and Greedy-Dual works better on byte hit rate and hit rate metrics.

Hao-Ping Hung [81] et al designed the shortest-path-based cache replacement for transcoding proxy systems. Unlike the cache replacement for conventional web objects, to replace some elements with others in the cache of a transcoding proxy, the method considers the transcoding relationship among the cached items. Replacement with Shortest Path (RESP) framework is developed to maintain the transcoding relationship and to perform cache replacement. The RESP framework contained two primary components such as procedure Minimum Aggregate Cost with Shortest Path (MASP) and algorithm Exchange-Based Replacement (EBR). Procedure MASP maintains the transcoding relationship using the shortest path table, whereas algorithm EBR performs cache replacement according to an exchanging strategy.
Keqi Li [82] et al proposed an efficient cache replacement algorithm which considers both the aggregate effect of caching multiple versions of the same multimedia object and cache consistency. They define a generalized aggregate cost saving to determine the rule for evicting the cached objects to make room for a new object if necessary. The aggregate cost is calculated based on the read cost, write cost, update cost and cache consistency of an object.

Kyungbaek Kim [83] et al proposed a least popularity-per-byte replacement (LPPB-R) algorithm for a proxy cache. The LPPB-R algorithm is a function-based algorithm which uses the popularity value of the object from proxy caches directly. This algorithm considers the variable size of the objects, since the web objects are not homogeneous. The function of the LPPB-R is to make the popularity per byte of the outgoing objects to be minimum value. Namely, when an object is to be inserted into the cache, the LPPB-R calculated the utilization of each object by using the popularity and the size. Then, LPPB-R evicted the object, which had the smallest utilization. Podlipnig and Böszörmenyi [86] analyzed various web cache replacement strategies. They classified cache replacement into four major categories of strategy: recency-based, frequency-based, function-based, and randomized. The adaptive replacement approach combines the strategies based on recency, frequency, and function, thereby yielding superior multimedia caching.

The study done by Lee [87] et al illustrated that the three levels of cache replacement as the key-frame level, prefix level, and segment level are applied. The study used the weighted LRU-k scheme, which considers the different priorities assigned to each level. The key frames and prefixes are deemed to be of high priority, due to poor consumption of cache resources. The segments are considered to be of low priority since the segments consumed more resources. In the weighted LRU-k caching replacement algorithm, objects of low priority are more likely to be removed, and objects of high priority are less likely to be replaced.

Debabala Swain [88] et al proposed an Adaptive-Weight-Ranking-Policy (AWRP), which assigned weight to each object and ranked each object as per the
weight. The weight is calculated after calculating the frequency index which holds the frequency of each block and the recency index which showed the time of the last access when it had been referenced against the total number of access to be made. After the weight is calculated, the least weight object is evicted from the cache.

A rank value (RV)-based replacement policy is developed by Gopalakrishnan Nair [89] et al for a multimedia server cache. In rank-based policy, the video objects are ranked based on the access trend by considering such factors as size, frequency, and cost. The video with the higher ranking is named “hot,” while the video with the lower ranking is named “cold.” The cold objects are replaced after determining the RV value of all objects.

Arlitt [93] et al. proposed a LFU-Aging in which the objects that are very popular during a time period can remain in the cache even when they are not requested for a long time period. This is due to their high frequency count. To avoid this cache pollution, an aging effect can be introduced. LFU-Aging uses, therefore, a threshold. If the average value of all frequency counters exceeds this threshold, all frequency counters should be divided by 2. Furthermore, this strategy uses a maximal value for the frequency counters.

Bahn [91] et al. developed a Least Unified Value (LUV) cache replacement algorithm in which the replacement strategy calculated for every object based on the ratio between the relative cost to fetch the object from its original server and is the probability that object is referenced in the future. Menaud [95] et al. described an LRU enhancement (LRU-QoS) that uses a separate LRU list for degradable objects and recompressable videos. Acharya [96] et al proposed to create different versions of an object, store them in the cache, and replace them independently. Under specific workload situations, this can give better results than simple cache replacement. Another proposal for a combination of cache replacement and transcoding can be found in Yeung [97] et al. Furthermore, there exist some proposals for combining transcoding and caching of streaming videos in heterogeneous client environments [98], [99].
A survey of web cache replacement strategies [100] is studied by Stefan Podlipnig et al. and proposed feedback approach constituted an intelligent way for adaptation but added some overhead to the implementation and performance of the replacement strategies.

Conversely, the prefix/suffix approach used the entire suffix as the replacement unit while segment sizes in the segment-based approach exponentially increase towards the end of the streaming multimedia object. The size of a processing unit must be flexible, as is the size of a cached fragment. This feature allowed the cache-replacement algorithm to adjust the amount of data cached for a streaming multimedia object in a finer granularity, so that, the proxy-cache space could be effectively managed.

2.7 Summary

The literature study is made on various parameters involved in the proxy caching such as type of proxy environments, measuring the startup latency, determining the size of the segments, choosing the caching part of the segments, updating the popularity of the objects and adjusting the caching part, adopting the random seek support and optimizing the cache space using various cache replacement policies.

The Table 2.1 summarizes the above parameters impact on proxy caching. The scalable proxy caching suitable to support heterogeneous environment clients with compromising the proxy caching performance in the fields of startup latency, random seek support, popularity maintenance, and increased processing complexity of the proxy server. Additional time required identifying the current network conditions and identifying exact versions or converting into the appropriate client requirement increases the startup latency and maintaining the popularity and user access patterns is tedious due to the keeping different versions.
Table 2.1 Comparison of various parameters on proxy caching

<table>
<thead>
<tr>
<th>Properties</th>
<th>Scalable Proxy Caching</th>
<th>Non-Scalable Proxy Caching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QoS</td>
<td>Layered encoding</td>
</tr>
<tr>
<td><strong>Bandwidth consumption rate</strong></td>
<td>Reduced</td>
<td>Reduced</td>
</tr>
<tr>
<td><strong>Startup Latency</strong></td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Cache space consumption rate</strong></td>
<td>High</td>
<td>Average</td>
</tr>
<tr>
<td><strong>Proxy jitter</strong></td>
<td>High</td>
<td>Average</td>
</tr>
<tr>
<td><strong>Bit-rate supported</strong></td>
<td>Variable Bit Rate</td>
<td>Variable Bit Rate</td>
</tr>
<tr>
<td><strong>Number of Objects and Versions</strong></td>
<td>Multiple versions</td>
<td>Single Version</td>
</tr>
<tr>
<td><strong>Content Adoption</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Cooperativeness</strong></td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td><strong>Cache processing complexity</strong></td>
<td>High</td>
<td>Average</td>
</tr>
<tr>
<td><strong>Popularity support</strong></td>
<td>Complex</td>
<td>Complex</td>
</tr>
<tr>
<td><strong>Random seek support</strong></td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td><strong>Replacement strategy support</strong></td>
<td>Complex</td>
<td>Complex</td>
</tr>
</tbody>
</table>

The non-scalable proxy caching is well supported in all the parameters as shown in Table 2.1, because the nature of the proxy server placement and the nature of the media cached in the proxy server. Compare with prefix/suffix caching method, the segment method provides well support on popularity maintenance and random seek support. Hence, our proxy caching method focuses on the segment-based proxy caching for the newly loaded objects with random seek support.