Chapter 11

CACHING AND DISTRIBUTION ISSUES FOR STREAMING CONTENT DISTRIBUTION NETWORKS

Michael Zink
Department of Computer Science
University of Massachusetts
Amherst, MA, USA
zink@cs.umass.edu

Prashant Shenoy
Department of Computer Science
University of Massachusetts
Amherst, MA, USA
shenoy@cs.umass.edu

Abstract This chapter presents an overview of the state of the art in caching and distribution techniques for streaming media content. A content distribution network (CDN)—an overlay network of proxy servers—is typically employed for this purpopse. We present techniques for caching entire files as well as caching partial content at each proxy in a streaming CDN. We then present techniques for designing a cooperative cluster of streaming proxies as well as techniques for streaming using peer-to-peer networks.

Keywords: Content distribution networks, streaming, caching

1. Introduction

1.1 Motivation

Content Distribution Networks (CDN) have become popular recently as a means to store web content closer to clients in a controlled manner. A content distribution network consists of an overlay network of proxy servers that are

geographically distributed and cache popular content close to clients; user requests for content are serviced by forwarding the request to the nearest proxy. Unlike pure on-demand caching approaches that only store content requested by clients in the proxy's cache, in a CDN, owners of web servers can also actively distribute (i.e. replicate) content to proxies. Today, CDN services are offered by numerous commercial companies (e.g., Akamai) to both content providers and end users. Content providers subscribe to CDN services to provide their clients with faster access to their content.

While the first generation of CDNs were designed primarily for web content, modern CDNs can also store and serve streaming media content such as audio and video. However, streaming media objects have different characteristics when compared objects such as text and images—these objects are several orders of magnitude larger in size and have larger bandwidth requirements. Consequently, unlike web content, naive replication of popular objects on all proxy servers in a CDN may not be efficient, and new techniques for distributing and caching streaming media objects need to be devised.

A content distribution network for streaming media will need to exploit the following characteristics of video content [Acharya and Smith, 1998, Acharya et al., 2000, Chesire et al., 2001]:

- Unlike web content which may be modified after its creation, video content follows the write-once-read-many principle. Thus, streaming CDNs are simpler in that they do not need to consider cache consistency issues.

- Popularity of video files follow the Zipf distribution, and each file is typically accessed sequentially. Caching and distribution techniques employed by the CDN need to be tailored for these characteristics.

- Due to the significantly higher storage space and bandwidth requirements as well as the timeliness constraints imposed on video accesses, distribution and caching techniques for streaming content need to designed with these constraints in mind.

- Internet broadcast or multicast may be employed by an origin server to deliver live content to a large number of users [Sitaram and Dan, 2000, Hu, 2001, Hua and Sheu, 1997, Paris et al., 1999, Eager et al., 2001]. Broadcast or multicast techniques may also be used to deliver very popular files to a large user population. Proxy servers within a CDN will need to support such broadcast and multicast techniques as well.

Despite some of these differences, streaming CDNs retain some of the key advantages of traditional web CDNs. First, since proxy servers replicate content from origin servers, content can still be delivered to end-users if the origin server or some of the proxy servers suffer a transient failure. Second, by placing

proxies close to the end-users and caching popular content at this proxies, a CDN can reduce the startup latency for accessing streaming content. Such caching also reduces the load on the origin servers and redistributes it across multiple proxies. Third, approximately 80% of the user requests access about 20% of the total available videos [Bianchi and Melen, 1997, Griwodz et al., 1997, Nussbaumer et al., 1995], indicating that, like in the web case, caching video content can be a very effective means for scalable delivery.

1.2 Outline

The remainder of this chapter is structured as follows. Section 2 describes the high level architecture of a streaming content distribution network. An overview on caching of complete objects is given in Section 3. Caching of partial streaming media objects is discussed in Section 4. Section 5 considers clusters of caches that are used to distribute the workload across several physical caches. Section 6 gives an overview of distribution networks based on peer-to-peer technology. Finally, Section 7 concludes this chapter.

2. Architecture of a Content Distribution Network

A typical streaming content distribution network is assumed to contain three components: origin servers, proxy caches and end-clients (see Figure 11.1). We briefly discuss the role of each component.

Origin Servers Origin servers store the original versions of each streaming media file. These servers are generally controlled by content providers. The content providers are assumed to have complete control over the content stored at origin servers—they can add new files to the server and delete old files. Further, they can also place restrictions on whether a file may be cached within the CDN. For reasons of simplicity, in this chapter,

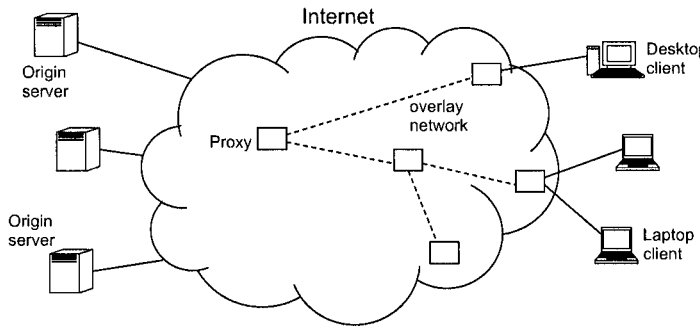

Figure 11.1. Architecture of a Content Distribution Network

we assume that no restrictions are placed on files and that any file may be cached within the CDN if it is advantageous to do so. In general, a typical CDN will serve content from multiple origin servers.

Proxy Caches A typical CDN will consist of a large number of proxy caches that are geographically distributed. Each proxy has a disk cache that is used to store streaming content from the origin severs. Proxies are assumed to be deployed close to the end-clients of the origin servers. A request for a streaming media object is typically forwarded to the nearest proxy. If the requested object or portions of it are already cached at the local disk, then the proxy can stream these portions to the end-client. The mission portions are handled by forwarding a request for these portions to another proxy or the origin server. Typically, the disk cache at each proxy is finite, and it is not possible to store all content from all origin servers at each proxy. Hence, each proxy is assumed to employ a caching policy that decides what objects or subsets of objects should be cached locally.

Clients End-users are assumed to request content via a client. Clients are assumed to be heterogeneous and may range from set-top boxes to standard PCs and from PDAs to mobile phones. Clients can have heterogeneous network bandwidth, processing power and display capabilities, and the origin server needs to tailor the requested content to meet the needs of its clients. Each client is assumed to make requests to its nearest proxy; such a proxy may be statically configured at the client, or dynamically determined by the CDN based on the client's location.

In the simplest case, a video (audio) file at the origin server consists of a sequence of frames (samples). Since such a file can impose significant storage space and bandwidth requirement, it is possible to partition each file in time or space. In the former case, each file is split into segments, where each segment contains a contiguous chunk of video and audio (see Figure 11.2(a)). For

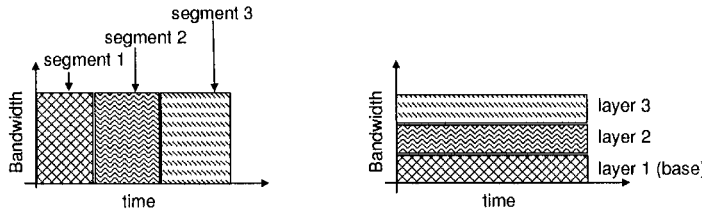

Figure 11.2. Unscalable vs. scalable content

instance, a one hour movie may contain three segments, containing the first five minutes, the next ten minutes and the final forty-five minutes of the movie, respectively. Each segment is smaller in size, when compared to the complete object, and can be handled independently by the CDN. An alternate approach is to partition the video file in the bandwidth (spatial) dimension. In this approach, each file is partitioned into a base layer and multiple enhancement layers (see Figure 11.2(b)). The base layer contains a coarse resolution version of each frame or sample, and each enhancement layer contains additional information to successively refine the resolution of data contained in the previous layers. Thus, layered encoding provides a multi-resolution representation of streaming content—to reconstruct the information included in layer n, all of the information of the lower layers (0, ..., n-1) are needed. While the notion of temporal partitioning into segments is independent of the compression format used to encode content, specific support is needed in the compression format to support layered encoding. Many modern compression algorithms such as MPEG-2 [MPEG-1, 1993], H.263+ [H263, 1995], MPEG-4 [Pereira and Ebrahimi, 2002] support layered encoding. In a later section, we will discuss how a CDN can exploit both temporal and bandwidth partitioning to support more effective distribution of streaming content.

With the above background, we discuss caching and distribution issues in streaming CDNs.

3. Complete Object Caching

In the simplest case, proxies within the CDN can cache entire audio or video files in their local disk cache. These files can be fetched from the origin servers on-demand as requests for these objects are received from clients, or by prefetching them *a priori*. In either case, since entire objects are cached at a proxy, due to their large sizes, only a limited number of files can be cached on disk. However, even this simple approach of caching entire objects can provide significant benefits, since caching very popular files locally can significantly reduce the burden on the origin servers.

Popularities of streaming media files tends to change over time as new files become popular and previously popular files see a reduced demand. Consequently, each proxy within the CDN needs to implement a caching policy that determines what files should be evicted and what new files to fetch into its cache. Typically, a *cache replacement policy* is employed to make such decisions. A common replacement policy that is widely used in web proxy caches is the *least recently used (LRU)* policy, where the least recently used object is evicted from the cache to make room for a new object. However, the LRU policy is not suitable for streaming media caches, due to the skewed popularities of these files. Studies have shown that the *least frequently used (LFU)* is more appropriate

in such situations. The policy maintains the frequency of access (a measure of popularity) for each file and evicts the object with the least access frequency (i.e., the least popular) object from the cache. Thus, more popular objects are given preference over less popular objects in the cache.

Although LFU is a better choice than LRU for proxies caching streaming media content, it suffers from two drawbacks. First, a previously popular object continues to be associated with a high frequency count and is not evicted from the cache even though it is no longer popular. A simple modification that overcomes this drawbacks is to decrease the frequency count over time, so that the frequency count accurately reflects the current popularity of an object.

Second, depending on the encoded bit rate and the length of the clip, different audio and video files will have different sizes. Since LFU does not take the size of the file into account, two files with vastly different sizes and identical popularities see identical treatment. To overcome this drawback, the caching policy needs to normalize the popularity of an object by its size. The resulting policy is referred to as *bandwidth to space ratio (BSR)* based caching [Tewari et al., 1998]. In this policy, the proxy computes the ratio of the bandwidth and space requirement for each file. The bandwidth requirement is a measure of popularity, since it is the total bandwidth required to service all concurrent, asynchronous requests for a file. The space requirement is simply the storage space needed to stored the file in the cache. The BSR policy caches objects with the largest BSR ratio, thereby preferring objects with higher popularities and smaller sizes.

4. Partial Caching of Video Objects

Since the large size of streaming media objects limits the utility of complete object caching, several policies that cache portions of streaming media files have been proposed in the literature. These policies assumes that each streaming media file is partitioned in time (into segments) or in space (into layers), and subsets of these components are cached at proxies in the CDN. Caching partial objects allows greater flexibility and enables more judicious use of proxy resources in the CDN. When a client requests a file, the cached portions of the file are served from the local disk cache, while the remaining portions are fetched from another proxy or the origin server. The rest of this section discusses these techniques in detail.

4.1 Time-based Partial Caching

We discuss two techniques for caching portions of objects that are partitioned in time, namely prefix caching and caching of arbitrary segments.

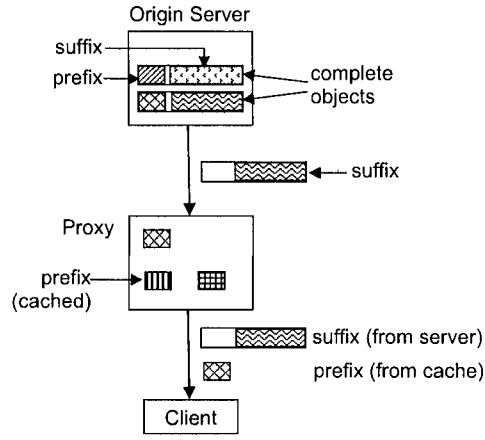

Figure 11.3. Prefix caching

Prefix Caching. In prefix caching, each video file is partitioned into two segments—a prefix and a suffix. The prefix contains the the initial portion of the video while the suffix contains the remainder of the file. Proxies in the CDN only cache prefixes and the suffixes are stored at the origin servers (see Figure 11.3). When a request arrives, the proxy can immediately start streaming the requested object if its prefix is cached locally, while fetching the remainder of the file from the server. Doing so can significantly reduce the startup delay seen by the client. The prefix also servers as a "buffer" at the proxy and enables the proxy to mask jitter and loss on the server-proxy network path. Prefix caching is motivated by the observation that an initial portion of a video file is more likely be viewed than latter portions. This is because many users start viewing an audio or video file and then decide to no longer view the remainder of the file. In such a scenario, caching entire files at the proxy may be wasteful; instead it is more useful to store prefixes of popular files at proxy servers.

A key parameter in prefix caching is the size of the prefix cached at the proxy. The prefix length must be chosen carefully to balance the storage space at the proxy and factors such as the file popularity, jitter and loss on the server-proxy path. The problem of optimal prefix cache size selection has been studied in the literature [Wang et al., 2002].

Caching of Arbitrary Segments. Prefix caching is the simplest time-based partial caching strategy, where each video file is partitioned into two segments, and proxies only cache the initial segment of a video file. The notion of caching partial file contents can be generalized to derive a variety of caching strategies, where a video file can be partitioned into multiple segments and different subsets

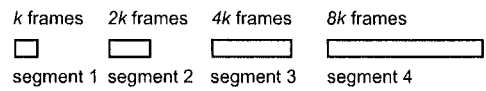

Figure 11.4. Exponential object segmentation

of segments may be stored at a CDN proxy. Numerous such segment caching strategies have been proposed in the literature.

One such scheme is to create segments of exponentially increasing sizes (see Figure 11.4), where each segment is twice the size of the previous segment [Wu et al., 2001]. The proxy computes the ratio of the access frequency and the segment number and caches segments with larger frequency to distance ratios. This strategy is similar to the bandwidth to space ratio (BSR) policy outlined earlier. The use of unequal segment sizes has two advantages. First, since viewers are more likely to watch initial portions of videos, the use of smaller segment sizes for the initial portions enables a proxy to favor these initial segments (over the subsequent larger segments) for caching. Second, unequal segment sizes enable a proxy to quickly free up a large amount of space by discarding a large segment that is no longer popular. In case of equal size segments, multiple evictions are necessary to free up the same amount of space. Consequently, it is easier for a proxy to adapt to changing segment popularities.

Another approach had advocated caching of the prefix as well as arbitrary segments of the video at the proxy, while storing the remainder of the file at the server. This approach is called selective caching [Miao and Ortega, 1999], and suggests storing intermediate segments or frames of the video at the proxy to permit efficient scan operations such as fast-forward, jump or rewind. Storing of the prefix allows the technique to provide similar benefits to prefix caching.

4.2 Bandwidth-based Partial Caching

In contrast to time-based partial cache, bandwidth-based partial caching requires that the video file be partitioned in the spatial dimension and involves storing portions of the file at proxies in the CDN and the remainder at the origin server. We discuss two techniques that are based on this idea.

Video Staging. The video staging approach [Zhang et al., 2000] involves partitioning the file into two layers—a fixed rate lower layer and a variable rate upper layer. The lower layer is stored at the server, while the upper layer is stored at one or more proxies in the CDN (see Figure 11.5). When a client requests a file, the lower layer is streamed from the origin server and the upper layer from a proxy cache, thereby reducing the backbone bandwidth requirement. The main advantage of video staging is that it imposes a fixed overhead on the

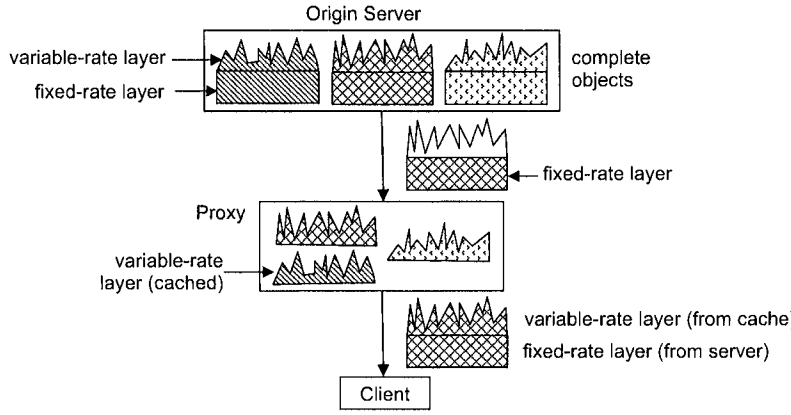

Figure 11.5. Video staging

server and the server-proxy network path due to the fixed rate of the lower layer. The variability in the video file is handled by the proxy. Observe that video staging is analogous to prefix caching, except for one important difference—in video staging, the lower layer is stored at the server, while in prefix caching, the prefix, which is the first segment, is stored at a proxy.

Caching of Layered Video. While bandwidth-based techniques for caching partial content have received little attention, some recent efforts have focused on caching of layered video files [Rejaie et al., 1999, Rejaie and Kangasharju, 2001a].

The Mocha proxy addresses this issue by assuming layered video files and caching different layers at a proxy, while fetching additional segments from the origin server in an on-demand basis [Rejaie et al., 2000, Rejaie and Kangasharju, 2001b]. User requests are serviced by streaming cached layers from the local disk, while fetching the missing layers from the origin servers (see Figure 11.6). A rate adaptive protocol is used to prefetch missing layers in the cache in a demand driven fashion in order to improve the quality of the cached video. Observe that, this technique is analogous to caching of arbitrary segments of the video at a CDN proxy. Other related efforts have also proposed caching of certain layers of a video file at CDN proxies and using user-perceived quality metrics to dynamically fetch additional layers from the origin server in order to improve video quality [Zink et al., 2004, Zink et al., 2003].

A disadvantage of partial time- or bandwidth-based caching techniques over caching complete objects is the reduced resilience to failures. Since only a subset of the file is stored in the CDN, the remaining content needs to be fetched from the server. Thus, a server failure will result in the content becoming

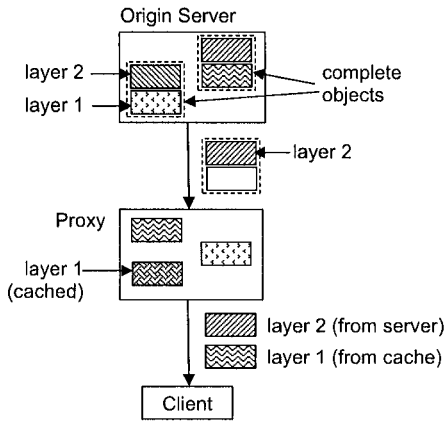

Figure 11.6. Caching of layered video.

unavailable to the end-user. In contrast, caching complete objects enables the CDN to mask server failures from the end-user.

5. Cluster-based Proxy Servers in a CDN

Our discussion thus far has focused on caching techniques at a single proxy in a CDN. However, in many scenarios, the CDN consists of multiple proxy clusters, where each cluster contains multiple proxy servers interconnected by a high-speed LAN (see Figure 11.7). In these environments proxies in the cluster can cooperate with one another to cache content; such cooperation often results in better service to the end-user. For instance, the cache at each proxy can be shared with other proxies in the cluster—in the event of a cache

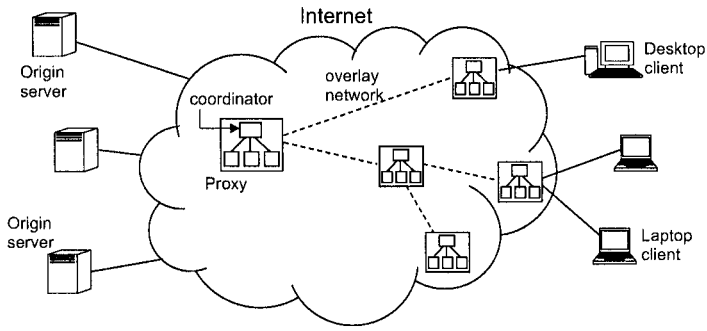

Figure 11.7. Architecture of a clustered-based proxies in a CDN.

Table 11.1. Summary of Cluster-based Proxy Architectures.

Approach	Architecture	Content	Coordinator
Intelligent Agent	Centralized, LAN	Layered	Yes
Middleman	Centralized, LAN	Segments	Yes
Rcache/Silo	Distributed, LAN	Segments	No
Dynamic Reconfiguration	Distributed, LAN	Segments	No
SOCCER	Distributed, WAN	Segments	No

miss, a proxy can simply forward a user request to another proxy in the cluster, rather than fetching the missing content from the origin server. Similarly, proxy clusters can be employed for load balancing, where multiple proxies in a cluster can participate in servicing requests for very popular videos. In this section, we discuss several cluster-based proxy architectures that have been proposed recently. Table 11.1 summarizes these approaches.

Middleman Cluster Proxy: Early work on cluster-based proxies assumed a centralized coordinator for proxies in the cluster. The coordinator is assumed to have full knowledge about the files cached at each proxy as well as the requests currently served by each proxy in the cluster. The coordinator is responsible for implementing the caching policy on behalf of all proxies in the cluster as well as for determining which proxy should service an incoming request. The Middleman cluster proxy is an example of such an approach [Acharya and Smith, 2000]. Middleman employs a coordinator proxy to make decisions for all proxies in a cluster. Upon receiving a request, a proxy forwards the request to the coordinator, which determines the best proxy in the cluster to service the request. The requested file is then streamed from that proxy to the end-client. In the event of a cache miss or if all proxies are overloaded, the request is forwarded to the origin server. Middleman also implements an *LRU-k* cache replacement strategy.[1] Each video file is partitioned into equal size segments and the LRU-k policy is employed to determine which segments are cached locally by each proxy in the cluster.

While Middleman requires that a file be partitioned into segments and employs time-based partial caching within the proxy cluster, caching of layered video in a proxy cluster has also been studied in the intelligent agent cluster-based proxy [Paknikar et al., 2000]. Like in Middleman, a centralized coordinator is used to determine which layers of a video should be cached and at which proxy in the cluster.

Silo and RCache Cluster-based Proxy: Since a centralized coordinator can become a bottleneck and is a single point of failure, distributed architectures for proxy clusters have been studied recently. Such cluster-based proxy do not assume the presence of a coordinator and make all decisions in a distributed fashion.

The RCache approach employs a cluster of K cooperating proxies that randomize the placement of video segments onto proxies in the cluster [Chae et al., 2002]. RCache proxies assume equal size segments for a video and cache a segment with a fixed probability a/K, where a is a constant. A randomized strategy that maps video segments to proxies can avoid hot spots that occur when multiple popular segments are placed on a single proxy, resulting in overload. Note, however, that in the RCache approach, the probability of caching a segment is independent of its popularity.

The Silo approach overcomes this drawback by using unequal segment sizes and a randomized placement strategy that takes popularities into account [Chae et al., 2002]. The approach also assumes a cluster of proxies and employs exponentially increasing segment sizes and exponentially decreasing probabilities for caching later segments of a video. Specifically, the i^{th} segment of a video has a size α^{i-1} and is cached with a probability $1/\beta^{i-1}$. Thus, initial segments are more likely to be cached (the first segment, also the smallest, is cached with a probability 1). Since different files can have different popularities, the probability of caching a segment can be biased by its current popularity. The approach also uses a cache replacement policy that takes both the local popularity of a segment (i.e., the popularity at a single proxy in the cluster) and the global popularity (the popularity across all proxies) when evicting segments.

Dynamically Reconfiguring Proxy Cluster: Since popularities of video files change dynamically over time, the set of segments cached at proxies in a cluster needs to be adapted to this changing popularity. Specifically, video segments that are increasing in popularity need to be fetched into the caches and segments that are no longer popular need to be replaced. A cluster-based proxy that can adaptively reconfigure the caches at proxies in the cluster was proposed in [Guo et al., 2003]. In this approach, a video file is partitioned into segments, that are ranked according to their *bandwidth to space ratio (BSR)*. The mapping of segments onto caches in the cluster is modeled as a two-dimensional knapsack problem, and a greedy first-fit heuristic is employed to place these segments onto proxies in the cluster. Depending on its popularity, each segment can be replicated at multiple caches, placed on a single proxy cache, or not cached at all. As the popularity of video file changes over time, the proxy cluster adapts this initial placement in an incremental fashion to match the new object popularities. This adaptation involves (i) determining an ideal mapping for the new popularities and (ii) use of a minimum weight perfect matching (MWPM) heuristic that transforms the current mapping to the new mapping such that the overhead of moving segments from one cache to another is minimized. Like in the Silo and the Middleman approaches, each user request is serviced by streaming available segments from the proxy cluster, while fetching the missing segments from the origin server.

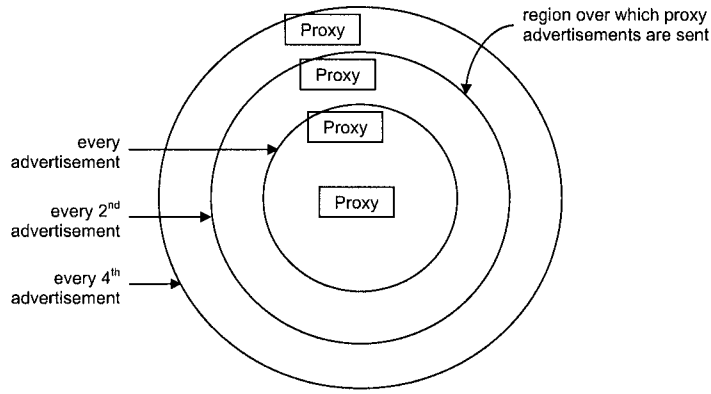

Figure 11.8. The expanding ring advertisement protocol.

Self-Organizing Cooperative Caching Architecture (SOCCER): While Middleman, Silo and the dynamically reconfiguring proxy architectures assume a cluster of proxies interconnected by a local area network, the SOCCER approach extends the notion of a proxy cluster to a wide area network [Hofmann et al., 1999]. In this approach, a collection of geographically distributed proxy servers cooperate with one another to service user requests. Cooperative caching requires a proxy to know the current status of other proxies in its cluster. While maintaining the state of all proxies is easy in a LAN setting, exchanging state updates is expensive in WAN environments. The SOCCER approach explicitly addresses this issue using an Expanding Ring Advertisement (ERA) protocol, where a proxy multicasts any updates to its cache with dynamic TTL values. Proxies that are further apart receive such advertisements less frequently, while nearby proxies have more up-to-date knowledge of each other's proxy cache contents (see Figure 11.8). The advertisements are also used to propagate load information to other proxies. The knowledge of a proxy load and its cache contents enables other proxies to determine whether to forward a user request to it.

6. Peer-to-Peer Streaming Techniques

The previous sections have discussed streaming from a client-proxy-server perspective. In this section, we provide an overview of an alternative streaming approach, namely peer-to-peer streaming. Peer-to-peer streaming is motivated by a number of reasons. Origin servers as well as CDN proxies have limited capacity on the number of concurrent streams they can support. If an overloaded server or proxy is unable to service a client, then the client can request an object from another client (i.e., a peer) that is currently being served by the proxy. The

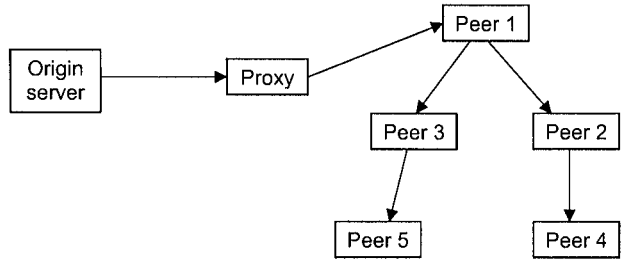

Figure 11.9. Peer-to-peer streaming

client then receives a stream from this peer and in turn streams the file to other interested clients. In other words, the proxy streams the file to a small number of client (or peers). Each peer in turn streams the file to other peers, until all interested clients receive a copy of the stream (see Figure 11.9). In effect, peer-to-peer streaming uses a form of multicast at the application level (referred to as application-level multicast). The network of peers form a multicast distribution tree rooted at the origin server or a CDN proxy. Observe that clients are inherently unreliable, since they can be disconnected or switched off by their owners at any time. Consequently, a peer-to-peer streaming technique must be able to adapt the distribution tree as peers dynamically join and leave the network. Further, the distribution tree needs to employ redundant streaming techniques to mask disruptions in the media playback caused by peers leaving the network. Consequently, much of the ongoing research in P2P streaming focuses on: (i) tree construction algorithms, (ii) handling dynamic joins and leaves, and (iii) redundant streaming techniques that mask packet losses due to peers leaving the network. This section discusses several peer-to-peer streaming approaches that have been proposed in the literature.

Zigzag streaming: The Zigzag P2P approach focuses on tree construction algorithms for streaming content to a large number of peers. The proposed tree construction causes the depth of the tree to grow logarithmically with the number of peers [Tran et al., 2003]. A distribution tree with a smaller depth reduces the number of intermediaries between the server and a leaf peer, which in turn reduces the end-to-end delay. The technique also bounds the number of children for each intermediate node in the tree, which in turn limits the network bandwidth requirement at each peer. Reducing the bandwidth requirements at each peer is especially important, since many peers tend to be connected over home broadband connections, which are asymmetric in nature with limited upload bandwidth. Departures of a peer are handled by reconnecting all peers in the subtree rooted at the departed peer with other live peers in the system.

CoopNet Resilient P2P streaming: The resilient P2P streaming approach [Padmanabhan et al., 2003] focuses on the use of redundant network paths (in the form of multiple distribution trees) and redundant data transmissions to minimize the disruptions due to peers leaving the network. A peer-to-peer streaming system called CoopNet ("cooperative networking") is proposed that employs multiple descriptive coding. Multiple description coding (MDC) is a method of encoding an audio and/or video stream into M separate streams, or descriptions, such that any subset of these descriptions can be received and decoded. The distortion with respect to the original signal is proportional with the number of descrip- tions received; i.e., the more descriptions received, the lower the distortion and the higher the quality of the reconstructed stream. This differs from layered coding in that in MDC every subset of descriptions must be decodable, whereas in layered coding only a nested sequence of subsets must be decodable. By using MDC to encode a video stream and by sending it on multiple paths to each peer, the CoopNet system ensures that a peer can decode the video so long as it receives some subset of the descriptions. The use of MDC also enables CoopNet to handle peer departures in a flexible manner—a departed peer results in a lower resolution video for all peers served by it until the tree is repaired.

SplitStream: The SplitStream system [Castro et al., 2003] combines a peer-to-peer overlay network with an application level multicast for the purpose of video distribution. In their approach a video is split into stripes which are distributed via separate multicast trees. The goal of SplitStream is to create a forest of separate multicast trees in such a way that a node is only an internal node for only one multicast tree and a leaf node in all other cases. This mechanism distributes the load equally over all nodes of the distribution system. This approach is well suited for the distribution of layer-encoded video, since each layer can be distributed via a separate multicast tree.

Layered Peer-to-Peer Streaming: The issue of streaming layered video in a peer-to-peer fashion has been explicitly studied in [Cui and Nahrstedt, 2003]. In this approach, each video is assumed to be encoded into a number of layers, and peers are assumed to be heterogeneous. An algorithm to determine what layers should be fetched by a peer from upstream peers and which of these layers to buffer and forward to downstream peers is proposed. The algorithm attempts to judiciously use the download and upload bandwidth available to each peer with the goal of maximizing overall viewing quality. For instance, a peer may fetch six layers from upstream peers, and depending on how many downstream peers it serves and when they join, only a subset of these six layers may be forwarded to others.

PROMISE P2P Streaming: The PROMISE peer-to-peer streaming system focuses on a set of services that are necessary to judiciously construct a P2P

distribution tree [Xu et al., 2002, Hefeeda et al., 2003]. For instance, PROMISE supports a service to infer the underlying network topology between peers. The knowledge of the underlying topology can be employed to construct a tree where the number of hops between two connected peers is small. Another service supported by PROMISE involves monitoring the status of peers, so that peer departures can be detected with low latency. Detecting peer failures quickly is important for minimizing the disruptions in video delivery to downstream peers. A third service involves failure recovery, where the service dynamically switches peers to standby peers, when a parent peer fails. Together, these collection of services simplifies the building of peer-to-peer streaming services over a wide area network. This was demonstrated by designing a multi-path P2P streaming using these services [Xu et al., 2002].

7. Conclusions

In this chapter, we presented a brief overview of the state of the art in caching and distribution techniques for streaming media content. Initial work in the area focused on full object caching. Given the large sizes of streaming media files, techniques for partial caching of objects have also been extensive studied. We classified partial caching techniques into two types: time-based partial caching and bandwidth-based partial caching. More recent research efforts have focused on caching techniques for a co-located cluster of proxy caches as well as techniques for streaming using peer-to-peer networks. The area continues to evolve as researchers shift their attention to distributing streaming content to small networked devices such as PDAs and next-generation mobile phones.

Notes

1. The *least recently used-K (LRU-K)* policy maintains a history of the K most recent access times and computes the K-distance of an object as the difference between the current time and the K^{th} access. The object with the greatest K distance is evicted from the cache. LRU-k is a generalization of the basic LRU policy, where LRU-1 is same as LRU.

References

Acharya, Soam and Smith, Brian (1998). Experiment to Characterize Videos Stored on the Web. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), San Jose, CA, USA*, pages 166–178.

Acharya, Soam and Smith, Brian (2000). MiddleMan: A Video Caching Proxy Server. In *Proceedings of NOSSDAV 2000, Chapel Hill, NC, USA.*

Acharya, Soam, Smith, Brian, and Parnes, Peter (2000). Characterizing User Access To Videos On the World Wide Web. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), San Jose, CA, USA*, pages 130–141. SPIE.

Bianchi, Giuseppe and Melen, Riccardo (1997). Non Stationary Request Distribution in Video-on-Demand Networks. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'97), Kobe, Japan*, pages 711–717. IEEE Computer Society Press.

Castro, Miguel, Druschel, Peter, Hu, Y. Charlie, and Rowstron, Antony (2003). SplitStream: High-bandwidth Content Distribution in Cooperative Environments. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA*, pages 103–107.

Chae, Youngsu, Guo, Katherine, Buddhikot, Milind M., Suri, Subhash, and Zegura, Ellen W. (2002). Silo, Rainbow, and Caching Token: Schemes for Scalable, Fault Tolerant Stream Caching. *IEEE Journal on Selected Areas in Communications*, 20, 7:1328–1344.

Chesire, Maureen, Wolman, Alec, Voelker, Geoffrey, and Levy, Henry (2001). Measurement and Analysis of a Streaming-Media Workload. In *Proceedings of USITS'02: The 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, CA, USA.*

Cui, Yi and Nahrstedt, Klara (2003). Layered peer-to-peer streaming. In *Proc. of International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '03).*

Eager, Derek, Vernon, Mary, and Zahorjan, John (2001). Minimizing Bandwidth Requirements for On-Demand Data Delivery. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):742–757.

Griwodz, Carsten, Bär, Michael, and Wolf, Lars C. (1997). Long-term Movie Popularity in Video-on-Demand Systems. In *Proceedings of ACM Multimedia Conference 1997, Seattle, WA, USA*, pages 340–357.

Guo, Y., Ge, Z., Urgaonkar, B., Shenoy, P., and Towsley, D. (2003). Dynamic cache reconfiguration strategies for cluster-based streaming proxies. In *Proceedings of the Eighth International Workshop on Web Content Caching and Distribution (WCW 2003), Hawthorne, NY.*

H263 (1995). ITU-T: Video Coding for Low Bit Rate Communication. International Standard. ITU-T Recoomendation H.263.

Hefeeda, M., Habib, A., Botev, B., Xu, D., and Bhargava, B. (2003). Promise: Peer-to-peer media streaming using collectcast. In *Proceedings of ACM Multimedia 2003, Berkeley, CA*, pages 45–54. ISBN:1-58113-722-2.

Hofmann, Markus, Ng, T. S. Eugene, Guo, Katherine, Sanjoy, Paul, and Zhang, Hui (1999). Caching Techniques for Streaming Multimedia over the Internet. Technical report, Bell Labs, Holmdel, NJ, USA.

Hu, Ailan (2001). Video-on-Demand Broadcasting Protocols: a Comprehensive Study. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01), Anchorage, AK, USA*, pages 508–517. IEEE Computer Society Press.

Hua, Kien A. and Sheu, Simon (1997). Skyscraper Broadcasting: A new Broadcasting Scheme for Metropolitan Video-on-Demand Systems. In *Proceedings of the ACM SIGCOMM '97, Cannes, France*, pages 89–100.

Miao, Zhourong and Ortega, Antonio (1999). Proxy Caching for Efficient Video Services over the Internet. In *Proceedings of the 9th Packet Video Workshop, New York, NY, USA*, pages 36–44.

MPEG-1 (1993). International Organization for Standardisation (ISO). Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s – Part 1: Systems. International Standard. ISO/IEC 11172-1:1993.

Nussbaumer, Jean-Paul, Patel, Baiju, Schaffa, Frank, and Sterbenz, James P. G. (1995). Networking Requirements for Interactive Video on Demand. *IEEE Journal on Selected Areas in Communications*, 13(5):779–787. ISSN 0733-8716.

Padmanabhan, V., Wang, H., and Chou, P. (2003). Resilient peer-to-peer streaming. In *Proceedings of IEEE Intl. Coference on Network Protocols (ICNP), Atlanta, GA*, pages 16–27. ISSN 1092-1648.

Paknikar, Shantanu, Kankanhalli, Mohan, Ramakrishnan, K.R., Srinivasan, S.H., and Ngoh, Lek Heng (2000). A Caching and Streaming Framework for Multimedia. In *Proceedings of the ACM Multimedia Conference 2000, Los Angeles, CA, USA*, pages 13–20.

Paris, Jehan-Francois, Long, Darell D. E., and Mantey, Patrick E. (1999). Zero-Delay Broadcasting Protocols for Video-on-Demand. In *Proceedings of the ACM Multimedia Conference 1999, Orlando, FL, USA*, pages 189–197.

Pereira, Fernando and Ebrahimi, Touradj (2002). *The MPEG-4 Book*. Prentice-Hall. ISBN 0-13-061621-4.

Rejaie, Reza, Handley, Mark, and Estrin, Deborah (1999). RAP: An End-to-End Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies 1999 (INFOCOM'99), New York, NY, USA*, pages 395–399.

Rejaie, Reza and Kangasharju, Jussi (2001a). Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming. In *Proceedings of the 11th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'01), Port Jefferson, NY, USA*, pages 3–10.

Rejaie, Reza and Kangasharju, Jussi (2001b). Mocha: A quality adaptive multimedia proxy cache for internet streaming. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video Port Jefferson, NY.*

Rejaie, Reza, Yu, Haobo, Handley, Mark, and Estrin, Debora (2000). Multimedia Proxy Caching for Quality Adaptive Streaming Applications in the Internet. In *Proceedings of the Nineteenth Annual Joint Conference of the*

IEEE Computer and Communications Societies 2000 (INFOCOM'00), Tel-Aviv, Israel, pages 980–989.

Sitaram, Dinkar and Dan, Asit (2000). *Multimedia Servers*. Morgan Kaufmann Publishers. ISBN 1-55860-430-8.

Tewari, Renu, Vin, Harrick, Dan, Asit, and Sitaram, Dinkar (1998). Resource-Based Caching For Web Servers. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), San Jose, CA, USA*, pages 191–204.

Tran, Duc T., Hua, Kien A., and Do, Tai (2003). ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In *Proceedings of the 22th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03), New York, NY, USA*, pages 1283–1292. ISSN 0743-166X.

Wang, B., Sen, S., Adler, M., and Towsley, D. (2002). Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution. In *Proceedings of the 21th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02), New York, NY, USA*, pages 1726–1735.

Wu, Kun-Lung, Yu, Philip S., and Wolf, Joel L. (2001). Segment-Based Proxy Caching of Multimedia Streams. In *Proceedings of the Tenth International World Wide Web Conference, Hong Kong, China*, pages 36–44.

Xu, D., Hefeeda, M., Hambrusch, S., and Bhargava, B. (2002). On peer-to-peer media streaming. In *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS 2002), Wien, Austria*, pages 363–371. ISSN 1063-6927.

Zhang, Zhi-Li, Wang, Yuewei, Du, David H.C., and Su, Dongli (2000). Prospects for Interactive Video-on-Demand. *IEEE/ACM Transactions on Networking*, 8(4):429–442.

Zink, Michael, Künzel, Oliver, Schmitt, Jens B., and Steinmetz, Ralf (2003). Subjective Impression of Variations in Layer Encoded Videos. In *Proceedings of the 11th IEEE/IFIP International Workshop on Quality of Service (IWQoS'03), Monterey, CA, USA*, pages 134–154. ISBN 3-540-40281-0.

Zink, Michael, Schmitt, Jens, and Griwodz, Carsten (2004). Layer-Encoded Video Streaming: A Proxy's Perspecive. *IEEE Communications*, 42(8):96–103.