Video Delivery Technologies for Large-Scale Deployment of Multimedia Applications

Kien A. Hua[1] *Mounir A. Tantaoui*[1]

School of Electrical Engineering and Computer Science University of Central Florida Orlando, FL 32816-2362, U.S.A
{kienhua, tantaoui}@cs.ucf.edu

Wallapak Tavanapong[2]

Computer Science Department
Iowa State University
Ames, IA 50011-1040, U.S.A
tavanapo@cs.iastate.edu

Abstract – Deployment of a large-scale multimedia streaming application requires an enormous amount of server and network resources. The simplest delivery technique allocates server resources for each specific request. This technique is very expensive and is not scalable to support a very large user community such as the Internet. Hence, the past decade has witnessed tremendous research efforts to facilitate cost-effective, large-scale deployment of multimedia streaming applications. In this paper, we describe three complementary research approaches: server transmission schemes using multicast, streaming strategies with Application Layer Multicast, and proxy caching techniques. We discuss pros and cons of these technologies and provide our observations on current business solutions.

I. INTRODUCTION

Next-generation networks will support transmission rates that are orders of magnitude higher than current rates. They will provide advanced services not currently available. In particular, the explosive increase in commercial usage of the Internet has resulted in a rapid growth in demand for video delivery techniques. They are underlying technologies for many new, exciting multimedia applications. For examples, on-demand home entertainment gives customers the freedom to view movies from remote sites at any time in the comfort of their own home; distance learning provides opportunities for students to take courses taught at remote locations according to their individual needs and time constraints; digital video library lets remote users research and view videos from a large video library; just to name a few.

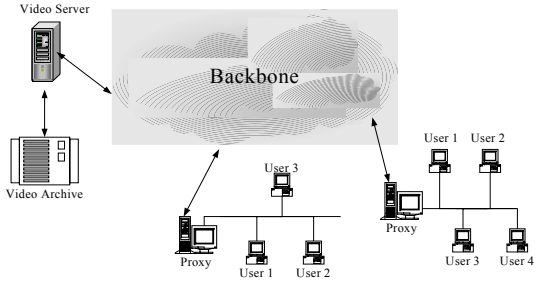


Figure 1. Video-On-Demand architecture

[1] This research is partially supported by the U.S National Science Foundation grant ANI-0088026
[2] and by the U.S National Science Foundation grant CCR-0092914.

Figure 1 depicts a simple architecture of a *Video-On-Demand* (VOD) system that consists of a video server with a video archive and a number of client machines connected via a local area network. Users use client software to request for their desired video. In response to a service request, the server delivers the requested video to the user in an isochronous data stream. The unit of server capacity required to support the playback of one video stream is referred to as a *channel*. The number of such channels is limited by the server network I/O bandwidth. The simplest delivery technique requires a dedicated server channel to serve each video request. Obviously, this scheme is excessively expensive and non-scalable. To conserve server network I/O bandwidth and wide-area-network bandwidth as well as to reduce service delays experienced by users, three complementary approaches have been investigated.

- **Server transmission schemes using multicast**: These techniques can be categorized into the **reactive transmission approach**, the **proactive transmission approach**, and the **hybrid approach**. In the reactive transmission approach, the server uses a few server channels to serve several requests for the same video arriving closely in time. This strategy allows the users to share server and network bandwidth. In the proactive transmission approach, clients do not make requests to the server. Instead, the server broadcasts a video periodically, e.g., a new stream of the same video is started every *t* seconds. The worst service latency experienced by any client is at most *t* seconds. A unique advantage of this approach is that it can serve a very large community of clients using minimal server bandwidth while guaranteeing a bounded service delay. In fact, the bandwidth requirement is independent of the number of concurrent clients using the system. The hybrid approach takes advantages of both the reactive and the proactive approaches. We will discuss the three approaches in more details along with other important issues such as supporting heterogeneous clients and *video-cassette-recorder-like* (VCR-like) interactivities.
- **Video streaming technologies with Application Layer Multicast:** In theory**,** IP multicast can be employed with the server transmission schemes. In practice, IP Multicast has deployment difficulty beyond a local area network [23]. As a result, several proposals for *Application Layer*

Multicast (ALM) have recently been introduced along with new video streaming techniques using ALM for live broadcast and pre-recorded videos. We discuss these techniques including their strength and weaknesses in this paper.

- **Proxy caching technologies:** A video proxy (a computer system equipped with large storage space) can be used to store popular videos close to the requesting clients (see Figure 1). The proxy delivers the cached portion of the requested video to the client while the remote video server needs transmit only the uncached portion of the video. This scheme minimizes the load on the wide-area-network and the video server. A well designed proxy caching technique can also reduce service delays and improve playback quality. We discuss several recent researches in proxy caching in this paper.

The remainder of this paper is organized as follows. In Section II, we discuss several transmission schemes using multicast. We present recent video delivery techniques using application layer multicast in Section III, and discuss proxy caching technologies in Section IV. We present our observation on the business applications of these technologies at the end of each section. Finally, we offer our concluding remarks in Section V.

II. Server Transmission Schemes Using Multicast

In this section, we first describe categories of video services, and then present techniques in the reactive transmission approach, the proactive transmission approach, and the hybrid approach. We conclude this section with discussion on related issues and relevant commercial video server software.

A. Categories of Video Services

Video services can be classified into the following categories based on the scheduling policies of data delivery and on the degree of interactivity [49,51].

- **No-Video-on-Demand:** This service is similar to the broadcast TV service where a user is a passive participant in the system and has no control over the video session. In this case, users do not request videos.
- **Pay-Per-View:** This service is similar to the cable TV Pay-Per-View. Users sign up and pay for specific services scheduled at pre-determined times.
- **True Video-on-Demand (TVOD):** True VOD systems allow users to request and view any video at any time, with full VCR capabilities. The user has a complete control over the video session. The simplest way to achieve TVOD is to dedicate each channel to every user in the system. However, this simplest scheme is very expensive.
- **Near Video-on-Demand (NVOD):** Users requesting for the same video are served using one video stream to minimize the demand on server bandwidth. The server is, therefore, in control of when to serve the video. VCR capabilities can be provided using many channels delivering the different requested portions of the same video requested by the different users.
- **Quasi Video-on-Demand (QVOD):** The QVOD [2] is a threshold-based NVOD. The server delivers a video when the number of user requests for the video is greater than a pre-defined threshold. The throughput of QVOD systems is usually greater than that of NVOD systems.

B. Reactive Server Transmission Approach

To conserve the server network I/O bandwidth requirement, two approaches, **static multicast and dynamic multicast**, have been studied. In the static multicast approach, a video server serves a batch of requests for the same video that arrive within a short period of time using one server channel. This approach is also known as *Batching*. All clients of the same batch receive the same data from the same multicast tree. The difference among the different schemes in this approach is the policy to select which batch to serve first when a server channel becomes available. The dynamic multicast approach extends the static multicast approach, allowing late coming requests to join a batch currently being served by extending the multicast tree to include the newly arriving client.

1) Static Multicast Approach

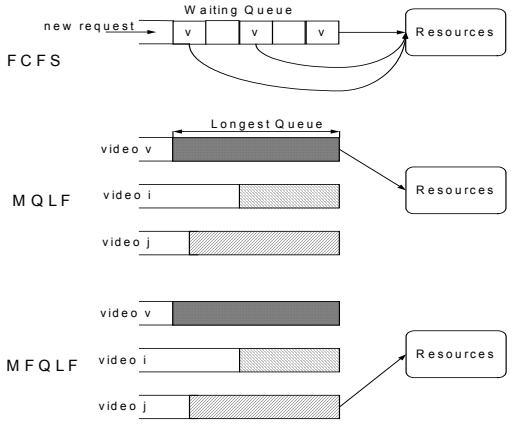


Figure 2. FCFS, MQLF, and MFQLF

Figure 2 depicts three static multicast schemes. In *First-Come-First-Serve (***FCFS***)*, as soon as some server bandwidth becomes free, the batch holding the oldest request with the longest waiting time is served next. In *Maximum-Queue-Length-First* (**MQLF**) [22], the batch with the most number of pending requests (i.e. longest queue) is chosen to receive the service. FCFS offers fairness since the scheme treats each user equally regardless of the popularity of the requested video. This scheme, however, yields low system throughput because it may choose to serve a batch with fewer requests first while another batch with more requests has to wait. To address this drawback, MQLF also maintaining a separate waiting queue for each video delivers the video with the longest queue (i.e., most number of pending requests) first. This policy maximizes server throughput, but is unfair to users who request less popular videos. *Maximum-Factored-*

Queued-Length First (**MFQLF**) [4] attempts to provide reasonable fairness as well as high server throughput. This scheme also maintains a waiting queue for each video. When a server channel becomes free, the policy selects the video v_i with the longest queue weighted by a factor $1/\sqrt{f_i}$ to be delivered, where f_i denotes the access frequency or the popularity of the video v_i. The factor f_i prevents the server from always favoring the popular videos. Since users have to wait in a queue to get the video data, it is important to note that the aforementioned batching policies can only provide a near-on-demand service.

2) Dynamic Multicast Approach

Unlike the static multicast approach, the dynamic multicast approach can offer true video-on-demand services while providing high throughput. This is achieved by letting late arriving requests for the same video to be serviced by dynamically expanding the already constructed multicast tree. In Adaptive Piggybacking [29], the server slows down the delivery rate of the video stream to a previous client, and speeds up the delivery rate of the video stream to a new client until they share the same play point in the video. At this time, the server merges the two video streams, and uses only one channel to serve the two clients. The adjustment of the delivery rate must be controlled within 5% to preserve the display quality of the video. This fact limits the number of channels Adaptive Piggybacking can merge to save resources.

Chaining [73] introduced the peer-to-peer streaming paradigm. By caching portions of the video data, clients can forward the video to other downstream clients, lessening the burden on the video server. In fact, client nodes form a delivery chain, and the server delivers the video through that chain using a single data stream. A new client either gets the video from an existing chain or from a new chain if the request cannot be served by the current chain. The main advantage of Chaining is that not all requests need be serviced from the video server and the video content is made available throughout the network at client nodes forming the chain. Subsequent schemes using a similar concept as Chaining include *Cache-relay-approach* [44] and *Proxy-based Asynchronous Multicast* [20]. The cache-relay approach allows the use of proxy storage in addition to the client storage. Proxy-based Asynchronous Multicast [20] allocates proxy cache space for its clients to store and forward data to the new clients requesting the same video at a later time. In this scheme, the dependency among the different arrival times of the requests for the same video is modeled by a dependency graph. Based on the dependency graph, a distributed algorithm is used to construct a minimum spanning tree to the proxies acting on behalf of the clients. The use of proxy storage makes Proxy-based Asynchronous Multicast more robust. Nevertheless, no actual performance comparison to Chaining was made.

Patching schemes [14, 17, 25, 37, 71] let a new client join an ongoing multicast and still receive the entire video data stream. For a new request for the same video, the server delivers only the missing portion of the requested video in a separate *patching stream*. The client downloads the data from the patching stream and immediately displays the data. Concurrently, the client downloads and caches the later portion of the video from the multicast stream. When finishing playing back the data in the patching stream, the client switches to play back the video data in its local buffer. Figure 3 illustrates an example of patching. It shows that users C_1 and C_2 are served together with a single multicast stream. At t time units later, user C_3 requests the same video. C_3 joins the multicast tree and starts buffering arriving video data, while at the same time C_3 downloads the missing portion via a *patching stream*. It is important to note that in patching, a user would get the video only if it is capable of simultaneously downloading from two streams (a regular multicast and a patching), and it has enough buffer space to absorb the time skew t. If these conditions do not hold, a new multicast stream is needed.

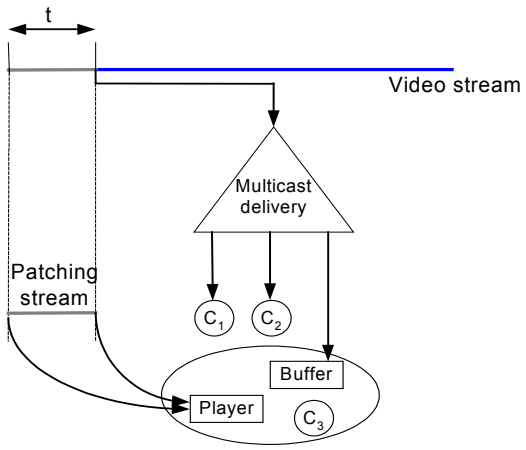


Figure 3. Patching

Users requesting the same video in the same *patching period t* are serviced using the same multicast stream. The appropriate choice of this period is essential to the performance of patching schemes. If the patching period is too large, there are many long patching streams, whereas, a small patching period results in many inefficient multicasts. In both cases, the advantage of multicast reduces. A patching scheme with an optimal patching period is presented in [14]. It minimizes the demand on server bandwidth. More recently, the patching concept has been extended to allow patching on the original patching streams [15]. That is, clients can download part of the missing portion from any earlier patching streams to reduce the patching cost. Double-Patching [15] tunes to no more than two streams at any one time. The limit of two streams is chosen because client bandwidth is generally scarce and expensive.

C. Proactive Transmission Approach or Periodic Broadcast

Several periodic broadcast schemes have been proposed in recent years [5, 21, 28, 35, 36, 38, 45, 52, 61, 62, 63, 64, 86].

In this approach, a video is fragmented into a number of segments. Each segment is periodically broadcast on a dedicated channel. A periodic broadcast system is highly scalable since it can serve a very large community of users requesting for the same video with minimal server bandwidth. The server bandwidth requirement is independent of the number of users the system is designed to support. The client tunes into one or more channels concurrently to download the broadcast segments into its buffer, while local playback software renders the video data in this buffer onto the screen. Periodic broadcast techniques guarantee that the client downloads a segment before its required playback time.

In this environment, if the client misses the beginning of the broadcast of the first segment, the client must wait until the next broadcast of the same segment. The worst service latency, therefore, is the time period between the consecutive broadcasts of the first video segments. Since the size of the first segment can be made very small, this approach can provide near-video-on-demand services. Many periodic broadcast techniques have been designed to keep the worst service delay small by making the first segment as small as possible while guaranteeing a jitter free playback at the client. Existing periodic broadcast schemes can be classified into two major categories, namely the *Server-Oriented Category* and the *Client-Oriented Category*. Techniques in the first category reduce service delays by increasing server bandwidth. On the contrary, techniques in the second category reduce the delays by requiring more client bandwidth.

1) *Server-Oriented Category*

Staggered Broadcasting [21] is the earliest and simplest video broadcasting technique. This scheme staggers the starting times for broadcasting a video evenly across available channels. The difference in the starting times is referred to as the *phase offset*. Because a new stream of the same video is started every phase offset, it is the longest time any client needs to wait for this video. The advantage of Staggered Broadcasting scheme is twofold. First, clients download data at the playback rate. Second, the clients do not need extra storage space to buffer the incoming data. This scheme, however, scales only linearly with the increase in the server bandwidth. *Pyramid Broadcasting* [86] addresses this drawback by broadcasting the video segments at a very high data rate, and allowing the clients to prefetch data into a local buffer. In this scheme, video segments are of geometrically increasing sizes, and the server network bandwidth is evenly divided to periodically broadcast one segment in a separate channel. This solution requires expensive client machines with enough bandwidth to cope with the high data rate on each broadcast channel. *Permutation Based Broadcasting* [5] improves this condition by dividing each channel into s sub-channels that broadcast a replica of the video fragment with a uniform phase delay. This strategy reduces the requirement on client bandwidth by some factor s although the data rate remains very high, which can still flood the prefetch buffer with half of the total data [36].

In *Skyscraper Broadcasting* [36], the server bandwidth is divided into *several* logical channels of bandwidth equal to the playback rate of the video. Each video is fragmented into several segments, and the sizes of the segments are determined using the following series referred to as the *broadcast series*; [1, 2, 2, 5, 5, 12, 12, 25, 25, …]. In other words, if the size of the first data segment is x, the size of the second and third segments are $2 \cdot x$, the fourth and fifth are $5 \cdot x$, sixth and seventh are $12 \cdot x$, and so forth. This scheme limits the size of the biggest segments (W-segments) to W units or $W \cdot x$. These segments stack up to resemble a skyscraper of a width W, thus the name Skyscraper Broadcasting. The server repeatedly broadcasts each segment on its dedicated channel at the playback rate of the video. To download the video, each client employs two synchronized threads - an *Odd Loader* and *an Even Loader*. They download the odd groups, each consisting of segments of odd size, and the even groups, each consisting of segments of even sizes, respectively. When a loader reaches the first W-segment, the client uses only this loader to download the remaining W-segments sequentially to minimize the requirement on the client buffer space. The main advantage of this technique is the fixed requirement on client bandwidth regardless of the desired service latency. To offer better service latency, one needs only add server bandwidth.

Client-Centric-Approach (CCA) [38] is a periodic broadcast technique that reduces the service latency by adding only server resources once clients' resources have been determined. In fact, CCA can be considered as a generalization of Skyscraper Broadcasting in that each transmission group can have more than two segments, and the number of data loaders is not limited to two as in Skyscraper Broadcasting. In contrast to the Skyscraper scheme, CCA can leverage extra client bandwidth, if available, to further reduce access latency.

2) Client-Oriented Category

The techniques in this category increase the requirement of both server and client bandwidth in order to reduce the service delay. *Harmonic Broadcasting* (HB) [45] initiates the techniques in this category. HB fragments a video into segments of equal sizes, and periodically broadcasts each segment on a dedicate channel. The channels, however, have decreasing bandwidths following the Harmonic series. In other words, the first channel is allocated a bandwidth equal to the playback rate of the video; the second channel has the bandwidth of half of the playback rate; the third channel has one third, and so forth. The client downloads segments from all channels concurrently. The original HB, however, cannot deliver all the segments on time. A simple delay, before consumption of the first segment, equal to the size of one segment solves this problem [61]. *Caution Harmonic Broadcasting* [61], *Quasi-Harmonic* and *Poly-Harmonic Broadcasting* [62] also address this problem. Although theses schemes use many channels to broadcast a video, the total bandwidth grows slowly following the Harmonic series, typically adding up to only *five* or *six* times the playback rate of the video. However, these schemes present another problem that involves the use of numerous channels (e.g., 240 channels are required for a 2-hour video if the latency is kept under 30 seconds). Since the client must concurrently obtain video data

segments from many channels, a storage subsystem with the capability to move their read heads fast enough to multiplex among so many concurrent streams would be very expensive. To solve this problem, *Pagoda Broadcasting* has been proposed [63]. This scheme also divides each video into segments of equal sizes. However, it addresses the problems of too many channels by broadcasting more than one segment on some channels. Pagoda Broadcasting does not require much more bandwidth compared with Harmonic Broadcasting and its variants and at the same time does not use as many channels.

The techniques in the Client-Oriented category have many drawbacks compared to those in the Server-Oriented category. First, the client must have network bandwidth equal to the server bandwidth allocated to the longest video. The requirement on the client bandwidth is therefore very high, making the overall system very expensive. Second, to improve access latency, it will require adding bandwidth to both server and client, which makes the system enhancement very costly. The justification for the Server-Oriented approach is that server bandwidth, shared by a large community of users, contributes little to the overall cost of the VOD environment. As a result, these techniques are less expensive than the Client-Oriented approach, which require a client to be equipped with substantially more client bandwidth. Nevertheless, if the bandwidth is readily available, these schemes or the CCA technique can be used. We note that CCA is not classified as a client-oriented approach because CCA allows the system to improve service latency by adding only server bandwidth as in Skyscraper Broadcasting.

C. Hybrid Broadcast Schemes

Periodic Broadcast techniques are most beneficial for popular videos. However, in general, these techniques are not applicable to less popular videos or a varying video access pattern. It was shown that a hybrid solution that combines both on-demand multicast and periodic broadcast offer the best performance [40]. *Adaptive Hybrid Approach* periodically measures the popularity of each video based on the distribution of recent service requests [40]. Popular videos are periodically broadcast using Skyscraper Broadcasting, and less requested videos are serviced using Batching. The number of channels used for periodic broadcast depends on the combination of popular videos currently in the system. The remaining channels are allocated to batching.

D. Other Important Issues

The reactive transmission schemes and periodic broadcast schemes facilitate large-scale deployment of video services. However, clients of such applications can use multiple types of receiving devices. Therefore, the dissemination of a homogeneous description of a video does not exploit high-bandwidth capabilities of some clients nor does it adapt to clients with low network bandwidth. A technique that can adjust to an array of heterogeneous receivers is required in practice. Furthermore, providing VCR-like capabilities (e.g., fast-forwarding or jumping to a specific frame in the video) is not only desirable but also required to quickly locate desired video content in some applications.

1) User-Heterogeneity

Multi-resolution encoding techniques [9] can be used to encode a video stream into a decomposition of layers referred to as *layered media formats* to serve heterogeneous clients. The lowest layer is referred to as the *base layer* and higher layers are referred to as *enhancement layers*. By delivering various layers in different multicast groups, each client can receive more or fewer layers depending on its bandwidth. This approach provides high adaptability, and is able to adjust to several bandwidth ranges. In *Receiver-Driven Layered Multicast* [54], a client keeps on adding layers until it observes a congestion. Higher layers may be dropped to alleviate the congestion. In other words, clients search for the optimal number of layers by trying to join and leave multicast groups.

In a periodic broadcast environment, even though a periodic broadcast scheme might be designed for a specific client bandwidth, a client, depending on its arrival time, might actually has less bandwidth than what is required by the broadcast scheme. *Heterogeneous Receiver-Oriented Broadcasting* [41] allows clients with different bandwidth to receive a video of the same quality from the same periodic broadcast. This scheme lets clients with high bandwidth download the video at the next occurrence of the first segment; but makes clients with less bandwidth wait for some specific time to start the download.

2) VCR-like Interactions

Techniques providing interactive services for the reactive transmission schemes have been introduced [7]. If the data in the prefetch buffer cannot service a forward or reverse jump request, the client requesting the interaction is served by another existing channel whose play point matches the client's destination point. If such a channel does not exist, the server issues an emergency channel to offer the service. The continuous actions such as fast forward or fast reverse, on the other hand, are supported by displaying the data in the client buffer first. Only when the needed frames are not in the buffer, the client switches to use an emergency channel.

Using emergency channels is expensive since one channel serves only one client. To reduce this cost, *Split and Merge* (SAM) protocol tries to move the client of an emergency channel to an on-going multicast whose play point is ahead of the client's play point, but not further than some amount [47]. In fact, SAM uses two types of channels, an *S* channel to support a normal playback of the video, and an *I* channel to provide interactive actions. When a client initiates an interactive operation, the client splits from its multicast group, and uses an *I* channel. After the completion of the interaction, the client merges the *I* channel with an existing multicast group. SAM protocol uses synchronized buffers to merge the clients. The drawback of this protocol is that it requires a tremendous number of *I* channels.

In the periodic broadcast environment, it was observed that the play point can be maintained at the middle of the video segment currently in the prefetch buffer in order to accommodate interactive actions in both forward and reverse directions [26]. *Active Buffer Management* (ABM) [27] carefully prefetches segments depending on the current position of the play point. ABM can also take advantage of the user behavior. If the user shows more reverse actions than forward actions, the play point can be kept near the end of the video segment in the buffer, and vice versa. The *broadcast-based interaction* technique improves the overall duration of an interaction by periodically broadcasting a compressed version of the video [75]. As an example, the compressed version of the video can consist of only one frame out of *f* frames. The process of watching the compressed segments at the playback rate has the effect of fast playing the normal video. This approach separates the client buffer space into two parts, the first part holds the normal version of the video, and the second part holds the compressed version. The two play points of the two separate buffers are held on the same frame throughout the download of the entire video. Since compressed data require less buffer space and download bandwidth, this scheme is able to support video interaction for a longer duration (e.g., fast-forward for a longer period of time).

3) Video Server Software

With the many designs of periodic broadcast schemes, a few software prototypes have been developed and experimented with. For specific periodic broadcast techniques, *Greedy-Disk Broadcasting* [12], *Striping-Broadcasting* [44], and a variant of Pagoda Broadcasting [77] have been implemented. A generalized periodic broadcast server (GPBS) software that supports many periodic broadcast techniques has also been developed [82]. Greedy-Disk Broadcasting, Striping-Broadcast, and GPBS implementation is based on IP-multicast. The common observation made from all the prototypes is that caching portions of broadcast videos in server memory improves the number of concurrent videos that can be supported by a broadcasting server. Our experience with GPBS on a Gigabit Ethernet indicates that to ensure a jitter-free broadcast from the server viewpoint, the Client-Oriented broadcast schemes such as Harmonic Broadcasting and its variants require much more server memory than those in the Server-Oriented category given the same server disk and network bandwidth. Given sufficient network bandwidth, the increase in server disk bandwidth has more impact than the increase in server memory in terms of supporting more broadcast videos for the broadcast schemes in the Server-Oriented category. Although these prototypes present a clear feasibility of the periodic broadcast approaches, they are not well suited for an unreliable environment such as the Internet where communication is very lossy. In fact, it was shown in the Greedy-Disk-Broadcasting [12] implementation that delivering a video from east to west of the United States can cause a packet loss factor up to 20%.

IP-multicast was originally designed for broadcasting situations where recovery is not needed for lost data. NACK-based and Tree-based protocols have been proposed to allow the receivers to acknowledge lost packets. In the Tree-based protocol, ACKs and NACKs are managed by the receiver's parent in a tree structure. The problem with these solutions is that the multicast efficiency decreases with the increasing number of receivers. This challenge has led many researchers to consider applying *forward-error correction* (FEC) to multicast. The main idea behind the use of FEC codes is to transmit the original source video data, in the form of sequence of some packets, as well as additional redundant packets, where the redundant information can be used to recover lost data packets at the receivers. Digital Fountain takes this approach in their periodic broadcast product [34]. It allows a server to periodically multicast streams of FEC encoded data packets. Clients tune into one or more multicast groups to receive the packets, and are able to reconstruct the original data using the FEC codes in the event of packet loss.

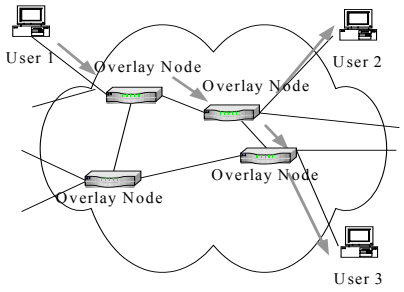
III. Video Delivery With Application Layer Multicast

In *application-layer-multicast* (ALM), end hosts implement multicast services at the application layer, assuming only IP unicast at the network layer. Existing application layer multicast protocols can be classified into two categories: **the infrastructure-based approach** and **the peer-to-peer (P2P) approach**. In the infrastructure-based approach, a set of dedicated machines called *overlay nodes* act as software routers with multicast functionalities [42]. Figure 4.a depicts an example of the infrastructure-based approach, where the multicast tree is constructed and packets are replicated at the overlay nodes. Video content is transmitted from a source to a group of receivers on a multicast tree comprising of only the overlay nodes. A new receiver joins an existing multicast group by connecting to its nearest overlay node. Due to the high cost of deployment and maintenance of the infrastructure-based approach, the P2P approach, i.e., Chaining, was introduced in [73], and has been studied intensively in recent years. The communication paradigm lets users' end hosts forward video data to other users' end hosts in the downstream. Figure 4.b shows the replication and forwarding of multicast packets by end-hosts or peers. In ALM, a packet may be sent on the same link more than once. Hence, this approach is less efficient than native IP Multicast. To quantify the effectiveness of ALM, two parameters, *stress* and *stretch,* are often used. The *stress* is defined as the number of the same packet sent over a specific link, and the *stretch* is the ratio of the path-length from the source to the destination peer via other peers to the length of the direct unicast path between the source and the destination peer.

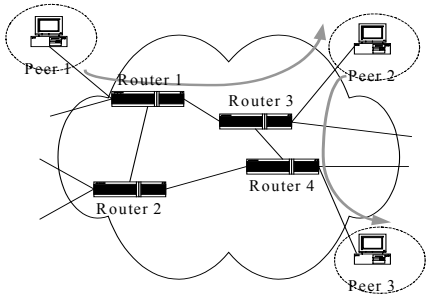
A. Infrastructure-based Approach

Range Multicast [39] utilizes an overlay structure consisting of overlay nodes placed at strategic locations on a wide-area network, and interconnected using unicast paths. As video packets pass through a sequence of overlay nodes on the delivery path, each node caches the video data into its fixed-size FIFO buffer. Such buffers, if they still have the first video frame, can be used to relay the entire video stream to

subsequent clients requesting the same video. The advantages of this technique are twofold. First, users experience no delay since they do not have to wait at the server for the batching period. Second, each multicast is very efficient because it can expand over time to accommodate many more users. This new capability is a fundamental shift from the conventional multicast concept where all members of a multicast group must share the same play point in the video stream at all times. Range Multicast gets its name from the fact that a single multicast can support a wide range of different play points simultaneously. Therefore, the data available from a range multicast at any time is not a "data point", but a contiguous segment of the video. The infrastructure-based approach lessens the bottleneck burden at the server side due to the fact that clients can get services not only from the server, but also from overlay nodes.



a. Infrastructure-based approaches



b. P2P communication at application level

Figure 4. Infrastructure-based vs. P2P

B. P2P Approach

The important design issues for video streaming in this environment are as follows.

- P2P streaming systems should offer a short access latency (quick joining time), allowing new peers to receive the desired video quickly.
- A quick and graceful recovery procedure is needed to handle peer failures. The failure recovery procedure should not only reconnect a disconnected peer to another peer, but must also quickly localize the failure such that only few peers are affected.
- Overhead for exchanging information among peers must be kept small.

Existing techniques in the P2P approach can be categorized into techniques supporting live video streaming and those that support pre-recorded video streaming. Some techniques can offer both services. Live video streaming differs from pre-recorded video streaming in two important aspects. First, the access latency is more crucial to live video streaming than to pre-recorded video streaming. Second, a user joining a current streaming session of live-streaming is only concerned about a stream starting from his/her joining time. Third, degradation of video quality for live video streaming is crucial since the option of watching the video for a second time may not be available [83].

1) P2P with Pre-recorded Video Streaming

Chaining [73] (described earlier in Section II-B) is the first P2P network. In Chaining, P2P concepts are applied for streaming pre-recorded videos. Each client in Chaining has a fixed-sized buffer to cache the most recent video data it receives. A new client can receive a video stream from another client that arrived earlier to the system as long as the latter still has the first block of the video data in its buffer. Since the original intent of Chaining was to reduce the video server burden by capitalizing a receiver's bandwidth to service other receivers, it does not provide a recovery protocol in the case of peer failures.

DirectStream [30] improves on Chaining by taking into consideration peers' bandwidth capacity for forwarding data (outbound-bandwidth to other peers). Peers in DirectStream include clients, content servers, and a directory server. The directory server acts as a central administrative peer for clients and other servers. To guarantee a smooth playback of a video for a peer after an early departure of its parent, DirectStream suggests buffering some amount of video data and delays the playback to deal with buffer starvation problems when such a disconnected peer takes a long time to locate a new parent. DirectStream has two drawbacks. The centralized management presents a single point of failure. When numerous different ancestors fail, a peer can quickly starve its buffer.

P2Cast [31] adapts the Patching concept (described in Section II-B) to the P2P environment. In P2Cast, a late coming client to the P2P system can receive a patch from other peers. This is achieved by forming a tree with sufficient bandwidth to transmit the stream taking into consideration that any peer can be used to deliver a patching stream. P2Cast requires each late coming client to download two streams, a base and a patching stream, simultaneously. Since P2Cast involves the source whenever a failure occurs, it is vulnerable to disruption due to server bottleneck at the source.

2) P2P with Live Streaming

Liveness of the video information is critical for many live streaming applications. This factor can be excessive in some P2P systems due to the latency in forwarding data over many peers. To keep this latency small, the tree height should be small. However, keeping the tree height as small as possible (e.g., all peers get their video data from the server) consumes more server bandwidth. Hence, in P2P live streaming, a

tradeoff between the height of the tree and the node degree should be taken into consideration.

Cooperative Networking or *CoopNet* [59] uses a multiple description coding method for media content, and can support both pre-recorded video streaming and live streaming. Multiple description coding is a way of encoding video signals into multiple separate streams such that subset of these sub-streams can be received and decoded into a signal. The sub-streams are delivered to the client through different peers. CoopNet constructs multiple distribution trees spanning the source and the receivers, where each tree delivers a single sub-stream. With multiple distribution trees, affected peers in the case of peer failures can still reconstruct the stream with fewer sub-streams. A drawback of CoopNet is that it requires a large amount of buffer space at each peer. In the case of pre-recorded video streaming, and if only partial video data is found at the serving peer, the requesting peer has to locate the missing part from other peers, which can incur more access delay. Another disadvantage is that the source has the overhead of maintaining knowledge of all distribution trees.

Peers' bandwidth heterogeneity are taken into consideration in [89]. In this scheme, multiple peers, each with limited outbound-bandwidth is used to provide video content to a receiving peer. First, the scheme computes the optimal media data assignment of peers delivering the content for each session. A session is therefore a many to many delivery. This leads to a minimum buffer delay encountered at the receiving peer. The authors also proposed a technique to amplify the capacity of a P2P streaming system. Capacity of a P2P system is defined in [89] as the total number of peer-to-peer streaming sessions that the system can simultaneously provide. Since requesting peers will eventually become supplying peers, the capacity of a P2P system is self-growing. A heuristic distributed admission control protocol using the classification of peers' outbound bandwidth is executed by both sending and receiver peers to achieve faster amplification and higher admission rate.

ZIGZAG [78] organizes a set of peers into a logical hierarchy of clusters of peers. Each cluster requires the number of peers to be in a bounded range and one leader elected from the peers in the cluster. Forming such a logical hierarchy has two advantages. First, the hierarchy is used to construct a multicast tree from the source to the different peers. Second, the hierarchy helps minimizing control message overhead and the number of affected peers when new peers join or existing peers leave. Figure 5 depicts a hierarchy of 32 peers including the server organized using ZIGZAG. Level 0 contains all peers that are clustered in eight groups of four. A higher level cluster contains only the leaders of the lower level clusters (e.g., a level 1 cluster contains four leaders of level 0 clusters). The arrows indicate the data delivery paths.

ZIGZAG uses the logical hierarchy and the following connectivity rules to construct a multicast tree that defines the data delivery paths.

A peer, when is not in its highest level, does not have a connection to or from any other peer. For instance, the server (S) at levels 1 and 0 does not have any incoming or outgoing links to other peers.

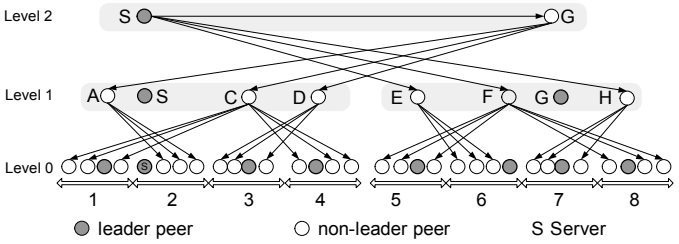


Figure 5. Hierarchy and multicast tree in ZIGZAG.

1) A peer, when in its highest level, can only connect to other peers from a different cluster in the immediate lower level (e.g., G at level 2 connects to A, C, and D).
2) Peers that are not leaders get data from a leader of a different cluster. For example, nodes F, E, and H in level 1 get data from S in level 2.

The technique is named ZIGZAG because the leader peer of each cluster does not forward the data to the peers in its cluster, but forwards it to the peers in a different cluster. If k is the minimum size of a cluster and N is the total number of peers, ZIGZAG guarantees the height of the tree to be $O(\log_k N)$ and a node degree $O(k^2)$. Keeping the height of the tree small improves the liveness of the video presentation while maintaining a bounded node degree minimizes the cost of reconnecting peers when some of their ancestor peers fails. Indirect ZIGZAG (I-ZIGZAG) is an extension of ZIGZAG [81], allowing non-leader peers to get their data from a co-leader in the same cluster. The co-leader, however, gets the data from a leader of a different cluster. For example, if applying I-ZIGZAG to the hierarchy in Figure 5, the cluster [A, Server, C, D] elects a co-leader, say C, different from the server who is the leader. C is responsible for getting data from upper levels, and forwarding it to peers in the same cluster and to co-leaders in clusters of lower levels. Peers A and D receive data from C in I-ZIGZAG instead of G in ZIGZAG. The two peers also forward data to co-leaders of lower levels. By changing the connectivity rules, allowing non-leaders to get data indirectly from a leader of a different cluster, the I-ZIGZAG multicast tree is longer, but peer degrees are smaller, which makes I-ZIGZAG more scalable to failed peers than ZIGZAG. However, ZIGZAG is more appropriate to live streaming due to its short multicast tree, which shortens the delay in video delivery.

3) P2P Streaming Software

There are several commercial P2P streaming products, such as Abacast [1], AllCast [6], and BlueFalcon [11], that create a tree structure based on receiver locations and connection types. However, there is no sufficient information for these systems for an exhaustive study. Most of these products have a centralized node for maintaining the entire tree, which makes them doubtful to work efficiently for a large group of transient receivers.

IV. VIDEO PROXY TECHNOLOGIES

Proxy caching has been shown effective to (1) reduce service delays, wide-area network load, and video server load, and (2) provide better playback quality. A proxy server is typically installed in the same local network as the clients. The proxy delivers the cached portion of the requested video to the client while the remote video server transmits only the uncached portion of the video to the client directly or via the proxy. We first discuss recent research in proxy caching technologies by dividing them into two broad categories: standalone proxy caching and collaborative proxy caching. Then, we present our observations on some existing video proxy software.

A. Standalone Proxy Caching

Recent researches in this category investigate various aspects of proxy caching, assuming a single video proxy.

1) Cache allocation policy

Cache allocation policy determines which portion of which video to cache at proxy storage. Existing cache allocation techniques can be divided based on server-proxy transmission schemes as follows.

a) With Reactive Transmission Schemes.

Prefix caching retains beginning portions (prefixes) of videos in the proxy [72]. This scheme reduces service delays since the client can play out frames in the prefix from the proxy while receiving subsequent frames (suffix) from the remote video server. Video staging [90] aims to reduce bandwidth variation of server-proxy data transmission. Video frames are divided spatially into two parts (instead of temporally as in prefix caching). The upper part typically has more variability in the bandwidth requirement and is stored at the proxy. The lower part has less variability in the bandwidth requirement and is transmitted from the video server when needed. Video staging achieves its design goal well. However, the technique does not significantly reduce service delays since the beginning of the second part of the requested video is needed from the remote server before the playback can begin. The synchronization between the two parts of the same video presents another practical problem.

Cache allocation policies that minimize the server-proxy transmission cost for different static and dynamic multicast schemes have been investigated [66]. Another work formulates cache allocation for layer-encoded videos as an optimization problem to maximize service provider's revenue given both the cache storage constraint and the server-proxy bandwidth constraint [46]. Three different heuristics were proposed. They are based on the popularity of the video layer, the revenue, and the revenue per required storage unit. The layer with the highest utility is allocated the cache space first. A layer is either cached entirely or not at all. Layers with lower utility values are considered next. When a lower layer of a video is not cached, all other higher layers of the same video are not cached since a lower layer is required to decode a higher layer.

A caching unit at a proxy can be an operating system block, an entire layer, a frame [50, 55, 76], a fixed-size segment, a segment of different predefined sizes [88], or a segment whose size is determined at runtime based on a history of access patterns [19]. While most caching techniques (with a frame as a caching unit) cache consecutive frames for the same video, the schemes in Reference [55] allow caching of non-consecutive frames at the proxy in addition to the prefix. For a network in which the server-proxy network bandwidth can be reserved, this scheme aims to minimize the server-proxy bandwidth and the client buffer requirement given limited proxy space for variable-bitrate videos. The idea is that larger frames that appear in various parts of a video if cached at the proxy can minimize the required server-proxy bandwidth since only the smaller frames need be transmitted from the server. For compressed videos, this scheme requires the proxy server to understand coding schemes in order to identify frame boundaries.

b) With Periodic Broadcast Schemes.

Caching the beginning frames (prefix) of very popular videos in a proxy can significantly reduce the server-proxy bandwidth for broadcasting the rest of the video [24]. This is because a fewer broadcast channels are needed to broadcast a shorter suffix. The important design issue is to ensure that the first segment of the suffix is available to the client before the entire prefix is played out. Hence, the first segment of the suffix is made equal to the size of the prefix [32]. Since videos are of different sizes and different bandwidth requirements, optimal prefix sizes are those that result in the minimum server-proxy bandwidth needed to broadcast the suffixes of the videos.

2) Cache replacement policy

Cache replacement policy determines which cache unit and how many of them to purge out when the current cache space is not enough to store the new video data. Several replacement policies including Least-Recently-Used variants have been investigated. Video sizes and popularity are often taken into account. The replacement policy proposed in Reference [43] is based on the idea that the proxy should favor caching data of popular videos that are difficult to get from the original video servers (i.e., the proxy-to-server path has limited bandwidth). This scheme keeps the video with high utility values in the proxy storage where the utility is defined as the ratio of the measured request rate of a video to the measured proxy-to-server bandwidth to the original video server storing the video. This scheme does not take into account the more advanced server-proxy transmission schemes.

Cache replacement policies for layer-encoded videos have also been introduced [68]. The policy in Reference [69] selects a victim layer - the layer with the lowest hit ratio. Segments in the victim layer are purged from the end and a new victim layer may be selected until enough space is obtained to cache the new data. The proxy also performs prefetching of segments not cached in the proxy by looking in a fixed-size prefetching window ahead of the current playout

time. The proxy issues a request for these segments, giving a higher priority to the missing segments in the lower layer. The prefetching technique is further extended by lengthening the prefetching window to the end of the video and by using different techniques to prioritize the missing segments 0. However, the amount of the storage space for the prefetched segments is not considered in this latter work.

B. *Collaborative Proxy Caching*

This approach takes advantage of aggregate network bandwidth and storage space of several proxies in the same Intranet to store more videos close to requesting clients. The number of participating proxies and their resources are known a prior. Proxy servers are either organized as a peer group [3] or a cache hierarchy [65]. Proxies as the leaf nodes of the hierarchy keep the prefixes of popular videos. Parents and siblings proxies are queried if the prefix of the requested video is not found in the leaf proxy responsible for the requesting client. Cache allocation in a peer group environment has been formulated as an optimization problem to minimize the server-proxy network bandwidth given both the storage constraint and the bandwidth constraint of the participating proxies [33]. In that study, the proposed cache allocation policies are based on the bandwidth-space heuristic as follows. Videos are sorted in a descending order according to the ratio of the bandwidth to the video size. Similarly, proxies in the peer group are sorted in a descending order of the ratio of the proxy network bandwidth to its cache space. The video objects are assigned to the proxies in the sorted order. Two additional techniques are introduced to handle changes in video popularity by swapping objects among proxies such that the exchanging overhead is minimized and workloads are balanced among the proxies.

Middleman [3] and the work in Reference [60] both use a centralized coordinator to perform all caching decisions in a peer group. The major difference between the two papers is the cache replacement policies. Middleman introduces a variant of LRU-k [58] as a cache replacement policy. Let *k-distance* of a video denotes the time difference between the current time and the time of the last k-access made to the video. The last block of the video with the largest k-distance is selected as the victim. Ties are broken by choosing the block from the least-loaded proxy. Other cache replacement policies such as a policy that takes into account access frequencies, video sizes, and times since the last access are investigated [60].

Instead of using proxies for video caching, Overlay Caching Scheme (OCS) [83] utilizes an infrastructure-based overlay architecture for caching. OCS is a distributed collaborative video caching scheme that caches one or more copies of the requested video in the caching overlay nodes along the path from the requesting client towards the video server. In OCS, locating the nearest cached copies is efficient, involving only a small set of caching overlay nodes. The number of cached copies per requested video is collaboratively controlled. Hence, the aggregated cache space is utilized more efficiently, which results in additional reduction in server load, network load, and service latency.

C. *Video Proxy Software*

Several video proxy products are available. They include Helix Universal Gateway from RealNetworks [67], Network Appliance NetCache [56], Novell Volera Media Excelerator [57], Certeon MediaMall [53], and BlueCoat ProxySG Series [10]. Since the underlying technologies of these products are not disclosed, we can only provide our observation based on the available information of these products. The common characteristics among them are the support of Realtime Streaming Protocol (RTSP) and well-known media players such as RealPlayer and Microsoft Windows Media Player. Other features mentioned by some of these products are load-balancing among proxies, the ability to setup a cache hierarchy, and the ability to set rules giving some group of users the ability to view high bandwidth videos.

V. CONCLUDING REMARKS

A picture is worth a thousand words, and the addition of sound and motion can breath life into a picture. As a result, video data have become an inseparable part of many applications with the rapid advances in networking technology. In particular, video on demand is a core technology for important applications such as digital libraries, distance learning, public information systems, electronic commerce, entertainment, just to name a few. The simplest video delivery technique employs a dedicated stream for each service request. Obviously, this scheme is too expensive and has little scalability. To vastly reduce this cost, one can leverage multicast technology to allow multiple clients to share a video stream. Unfortunately, today's multicast technology was developed in the 80's, and was not optimized for video applications. Missing a multicast could mean a long wait until the next multicast. This limitation has recently led to a large body of research looking for remedies at the application level. We discussed many of these solutions including some of our own in this paper.

Dynamic Multicast techniques such as *Patching* enable majority of the clients to receive most of their data from an existing multicast instead of demanding a whole new stream from the server. This strategy substantially reduces the demand on server bandwidth. For very popular videos, periodic broadcast techniques such as *Skyscraper Broadcasting* can be used to serve a very large user community using little server and network bandwidth. A few prototypes at universities and a commercial product from Digital Fountain have demonstrated the feasibility of this approach.

For environment where the multicast facility is not available, the peer-to-peer streaming approach, pioneered by the *Chaining* technique, offers a cost effective and highly scalable solution. Since peers can get videos from other peers, the load on the server is minimized. This concept has also been adapted for live video delivery, where peers forward video

streams to other peers in the downstream. A recent technique, called ZIGZAG, minimizes the height of the delivery tree to ensure good liveness of the video presentation while maintaining a bounded node degree to keep the cost of reconnecting peers low should some of their ancestor peers fail. *Range Multicast* is another interesting variation of the Chaining concept. Instead of relying on peers to forward data, this scheme deploys software routers at strategic locations in a wide-area network to facilitate data forwarding. The significance of this idea is the multicast of a sliding window over the video stream as opposed to a single data packet at a time as in conventional multicast. This new communication paradigm enables a single multicast to serve many clients with different play points in the same video. We have built a prototype to demonstrate this concept in a lab environment. Given the deployment difficulty of IP Multicast, Chaining-like techniques are promising and we start seeing some business solutions based on these technologies.

Besides video streaming techniques, we also discussed video proxy technologies in this paper. They are ready for actual deployment with quite a few commercial systems available. These systems have been shown to improve service delays, reduce network traffic, lessen server load, and provide better playback quality.

Although many great advances have been made on the Internet, none have had as great and direct an impact on the daily lives of ordinary people as video-on-demand applications. They are merging as an important development of this decade with the increasing use of all kinds of video data on the Internet. The techniques presented in this paper represent significant steps toward making video-on-demand technology ubiquitous.

REFERENCE

[1] "Abacast, Inc," http://www.abacast.com/.

[2] E. L. Abram-Profeta and K. G. Shin, "Scheduling Video Programs in Near Video on Demand Systems," In Proc. of ACM Multimedia'97, Seattle, WA, USA, November, 1997, pp. 359-369.

[3] S. Acharya and B. C. Smith, "MiddleMan: A Video Caching Proxy Server," In Proc. of NOSSDAV 2000, Chapel Hill, NC, USA, June, 2000.

[4] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On Optimal Batching Policies for Video-on-Demand Storage Servers," In Proc. of IEEE Int'l Conf. on Multimedia Computing and Systems, Hiroshima, Japan, June, 1996.

[5] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A Permutation-based Pyramid Broadcasting Scheme for Video-on-Demand Systems," In Proc. of Int'l Conf. on Multimedia Computing and Systems, Hiroshima, Japan, June, 1996.

[6] "Allcast," http://www.allcast.com.

[7] K. C. Almeroth and M. Ammar, "A Scalable Interactive Video-on-Demand Service using Multicast Communication," In Proc. of Int'l Conf. on Computer Communication Networks, 1994, pp. 292-301.

[8] K. C. Almeroth and M. Ammar, "On the Use of Multicast Delivery to provide a Scalable and Interactive Video-on-Demand Service," *IEEE Journal of Selected Areas in Communications*, vol. 14, pp. 1110-1122, August, 1996.

[9] A. Bayrakeri and R. M. Mersereau, "Temporally Scalable Video Coding Using Nonlinear Deinterlacing," In Proc. of the IEEE Data Compression Conference, March, 1997, pp. 423-423.

[10] "Blue Coat Systems, Inc," http://www.bluecoat.com.

[11] "Blue Flalcon Networks," http://www.bluefalcon.com.

[12] M. K. Bradshaw, B. Wang, S. Se, L. Gao, J. Kurose, P. Shenoy, and D. Towsley, "Periodic Broadcast and patching services - implementation, measurement, and analysis in an internet streaming video testbed," In Proc. of ACM Multimedia'01, Canada, October, 2001.

[13] H. M. Briceo, S. Gortle, and L. McMillan, "Naive-network aware internet video encoding," In Proc. of ACM Multimedia, Orlando, FL, USA, October, 1999, pp. 251-260.

[14] Y. Cai, K. A. Hua, and K. Vu, "Optimizing Patching Performance," In Proc. of ACM/SPIE Conf. on Multimedia Computing and Networking, San Jose, CA, January, 1999, pp. 204-215.

[15] Y. Cai and Kien A. Hua, "Sharing Multicast Videos Using Patching Streams," in *Multimedia Tools and Applications* journal, Vol. 21, No. 2, November 2003, pp. 125-146.

[16] Y. Cai, W. Tavanapong, and K. A. Hua, "Enhancing Patching Performance through Double Patching," In Proc. of Int'l Conf. on Distributed Multimedia Systems, September 24-26, 2003, pp. 72-77.

[17] S. W. Carter and D. D. E. Long, "Improving bandwidth efficiency of video-on-demand servers," *Computer Networks and ISDN Systems*, pp. 99-111, March, 1999.

[18] Y. Chawathe, S. A. Fink, S. McCanne, and E. A. Brewer, "A Proxy Architecture for Reliable Multicast in Heterogeneous Environments," In Proc. of ACM Multimedia, Bristol, UK, 1998, pp. 151-159.

[19] S. Chen, B. Shen, S. Wee, and X. Zhang, "Adaptive and Lazy Segmentation Based Proxy Caching for Streaming Media Delivery," In Proc. of NOSSDAV'03, Monterey, CA, USA, June, 2003.

[20] Y. Cui and K. Nahrstedt, "Proxy-based Asynchronous Multicast for Efficient On-demand Media Distribution," In Proc. of ACM/SPIE Multimedia Computing and Networking, San Jose, CA, USA, January, 2003, pp. 162-176.

[21] A. Dan and P. Shahabuddin, "Scheduling Policies for an on-Demand Video Server with Batching," In Proc. of ACM Multimedia'94, San Francisco, CA, USA, October, 1994, pp. 15-23.

[22] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic Batching Policies for an On-Demand Video Server," *Multimedia Systems*, vol. 4, pp. 112-121, June, 1996.

[23] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture", *IEEE Network magazine special issue on Multicasting*, vol. 14, 2000, pp. 78-88.

[24] D. L. Eager, M. C. Ferris, and M. K. Vernon, "Optimized Regional Caching for On-Demand Data Delivery," In Proc. of IS&T/SPIE Conference on Multimedia Computing and Networking 1999, San Jose, CA, January, 1999, pp. 301-316.

[25] D. Eager, M. Vernon, and J.Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," In Proc. of ACM Multimedia'99, Orlando, FL, USA, November, 1999, pp. 199-202.

[26] Z. Fei, I. Kamal, S. Mukherjee, and M. Ammar, "Providing Interactive Functions for Staggered Multicast Near Video-on Demand Systems," In Proc. IEEE Int'l Conf. on Multimedia Computing and Systems, June, 1999, pp. 949-953.

[27] Z. Fei, I. Kamel, S. Mukherjee, and M. Ammar, "Providing Interactive Functions through Active Client Buffer Management in Partitioned Video Broadcast," In Proc. of Int'l Workshop on Networked Group Communication, Pisa, Scuola Superiore S. Anna, Italy, November, 1999

[28] L. Gao, J. Kurose, and D. Towsley, "Efficient Schemes for Broadcasting Popular Videos," In Proc. of Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 98), Cambridge, UK, July, 1998.

[29] J. L. L. Golubchik and R. Muntz, "Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers," *ACM Multimedia Systems*, 1996, pp. 140-155.

[30] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "A Peer-to-Peer On-Demand Streaming Service and Its Performance Evaluation," In Proc. of IEEE Int'l Conf. on Multimedia and Expo (ICME'03), 2003, pp. 649-652.

[31] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast, P2P Patching Scheme for VoD Service," In Proc. of WWW, 2003, pp. 301-309.

[32] Y. Guo, S. Sen, and D. Towsley, "Prefix Caching Assisted Periodic Broadcast for Streaming Popular Videos," In Proc. of IEEE Int'l Conf. on Communications, Anchorage, Alaska, USA, May, 2003, pp. 2607-2612.

[33] Y. Guo, Z. Ge, B. Urgaonkar, P. Shenoy, and D. Towsley, "Dynamic Cache Reconfiguration Strategies for A Cluster-Based Streaming

Proxy," In Proc. of Int'l Workshop on Web Content Caching and Distribution, Hawthorne, NY, USA, October, 2003.
[34] G. B. Horn, P. Knudsgaard, S. B. Lassen, M. Luby, and J. E. Rasmussen, "A Scalable and Reliable Paradigm for Media on Demand," *IEEE Computer*, 2001, pp. 40-45.
[35] A. Hu, "Video-on-Demand Broadcasting Protocols: A Comprehensive Study," In Proc. of IEEE INFOCOM'01. Anchorage, Alaska, USA, April, 2001, pp. 508-517.
[36] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," In Proc. of ACM SIGCOMM'97, Cannes, France, September, 1997, pp. 89-99.
[37] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," In Proc. of ACM Multimedia'98, Bristol, UK, September, 1998, pp. 191-200.
[38] K. A. Hua, Y. Cai, and S. Sheu, "Exploiting client bandwidth for more efficient video broadcast," In Proc. of Int'l Conf. on Computer Communications and Networks, October, 1998, pp. 848-856.
[39] K. A. Hua, D. A. Tran, and R. Villafane, "Caching Multicast Protocol for On-Demand Video Delivery," In Proc. of ACM/SPIE Conference on Multimedia Computing and Networking (MMCN 2000), San Jose, CA, January, 2000, pp. 2-13.
[40] K. A. Hua, J.-H. Oh, and K. Vu, "An Adaptive Video Multicast Scheme for Varying Workloads," *ACM-Springer Multimedia Systems Journal*, vol. 8, pp. 258-269, August, 2002.
[41] K. A. Hua, O. Bagouet, and D. Oger, "A periodic broadcast protocol for heterogeneous receivers," In Proc. of ACM/SPIE Conf. On Multimedia Computing and Networking, San Jose, CA, USA, January, 2003.
[42] J. Janotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, "Overcast: Reliable multicasting with an overlay network," In Proc. of the 4th Symposium on Operating Systems Design and Implementation, San Diego, CA, USA, 2000, pp. 197-121.
[43] S. Jin, A. Bestavros, and A. Iyengar, "Accelerating Internet Streaming Media Delivery using Network-Aware Partial Caching," In Proc. of IEEE Int'l Conf. on Distributed Computing Systems, Vienna, Austria, July, 2002, pp. 153-160.
[44] S. Jin and A. Bestavros, "Cache-and-Relay Streaming Media Delivery for Asynchronous Clients," In Proc. of NGC'02, Boston, MA, USA, October, 2002.
[45] L. Juhn and L. Tseng, "Harmonic Broadcasting for Video on Demand Services," *IEEE Transactions on Broadcasting*, vol. 43, pp. 268-271, September, 1997.
[46] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross, "Distributing Layered Encoded Video Through Caches," In Proc. of IEEE INFOCOM, Alaska, USA, April, 2001, vol. 3, pp. 1791-1800.
[47] W. Liao and V. O. Li, "The split and merge (SAM) protocol for interactive video-on-demand," *IEEE Multimedia*, October-December, 1997, pp. 51-62.
[48] G. Lilienfield and J. Woods, "Scalable High-Definition Video Coding," In Proc. of IEEE Int'l. Conf. on Image Processing, Washington, D.C., USA, October, 1995, pp. 2567-.
[49] T. D. C. Little and D. Venkatesh, "Prospects for interactive video on demand," *IEEE JSAC*, vol. 14, pp. 1099-1109, August, 1996.
[50] W.-H. Ma and H. C. Du, "Reducing bandwidth requirement for delivering video over wide area networks with proxy server," In Proc. of Int'l Conf. on Multimedia and Expo, vol. 2, 2000, pp. 991-994.
[51] H. Ma and K. G. Shin, "Multicast Video-on-Demand Services," *ACM Communication Review*, pp. 31-42, January, 2002.
[52] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel, "Scalable On-Demand Media Streaming with Packet Loss Recovery," In Proc. of ACM SIGCOMM 2001, San Diego, CA, USA, August, 2001, pp. 97-108.
[53] "Media Mall," http://www.certeon.com/index.html
[54] V. J. S. McCanne and M. Vetterli, "Receiver-driven Layered Multicast," In Proc. of ACM SIGCOMM, CA, USA, August, 1996, pp. 117-130.
[55] Z. Miao and A. Ortega, "Scalable Proxy Caching of Video Under Storage Constraints," *IEEE Journal on Selected Areas in Communications*, vol. 20, September, 2002.
[56] "NetCache Product Family",http: //www.netapp.com/products/netcache/netcache_family.html.
[57] "Novell's Volera Media Excelerator," http: //www.novell.com/products/volera/media.html.
[58] E. O'Niel, P. O'Niel, and G. Weikum, "The LRU-k Page Replacement Algorithm For Database Disk Buffering," In Proc. of ICDE, 2003.
[59] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Distributing Streaming Media Content Using Cooperative Networking," In Proc. of Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'02), May, 2002, pp. 177-186.
[60] S. Paknikar, M. Kankanhalli, K. R. Ramakrishnan, S. H. Srinivasan, and L. H. Ngoh, "A Caching and Streaming Framework for Multimedia," In Proc. of ACM Multimedia'2000. CA, USA, October, 2000, pp. 13-20.
[61] J.-F. Paris, S. W. Carter, and D. D. E. Long, "Efficient Broadcasting Protocols for Video on Demand," In Proc. of Int'l. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOT'98), July, 1998, pp. 127-132.
[62] J.-F. Paris, S. W. Carter, and D. D. E. Long, "A Low Bandwidth Broadcasting Protocol for Video on Demand," In Proc. of Int'l Conf. on Computer Communications and Networks, Lafayette, LA, USA, October, 1998, pp. 690-697.
[63] J. F. Paris, "A simple low-bandwidth broadcasting protocol for video on demand," In Proc. of Int'l Conf. on Computer Communications and Networks, 1998, pp. 118-123.
[64] J.-F. Paris, S. W. Carter, and D. D. E. Long, "A Hybrid Broadcasting Protocol for Video on Demand," In Proc. of ACM/SPIE Multimedia Computing and Networking Conference (MMCN'99), San Jose, CA, January, 1999, pp. 317-326.
[65] Y.-W. Park, K.-H. Baek, and K.-D. Chung, "Reducing Network Traffic Using Two-Layered Cache Servers for Continuous Media Data on the Internet," In Proc. of IEEE Int'l Conf. on Computer Software and Applications, 2000, pp. 389-374.
[66] S. Ramesh, I. Rhee, and K. Guo, "Multicast with Cache (Mcache): An Adaptive Zero-Delay Video-on-Demand Service," In Proc. of IEEE INFOCOM, Alaska, USA, April, 2001, pp. 85-94.
[67] "RealNetworksProducts,"http://www.realnetworks.com/products/gateway/index.html.
[68] R. Rejaie, M. Handley, and D. Estrin, "Proxy caching mechanism for multimedia playback streams in the Internet," In Proc. of the 4th Int'l WWW Caching Workshop, March, 1999.
[69] R. Rejaie, H. Yu, M. Handley, and D. Estrin, "Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet," In Proc. of IEEE INFOCOM'2000, Tel-Aviv, Israel, March, 2000, vol. 2, pp. 980-989.
[70] R. Rejaie and J. Kangasharju, "Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming," In Proc. of NOSSDAV'01, Port Jefferson, NY, USA, June, 2001.
[71] S. Sen, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming," In Proc. of IEEE NOSSDAV'99, June, 1999.
[72] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," In Proc. of IEEE INFOCOM'99, New York, NY, 1999, pp. 1310-1319.
[73] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A Generalized Batching Technique for Video-on-Demand Systems," In Proc. of IEEE ICMCS'97, Ottawa, CA, 1997, pp. 110-117.
[74] S. Sheu and K Hua,"Scalable Technologies for distributed multimedia systems" PhD Dissert. SEECS. Univ. of Central Florida. 1999.
[75] M. Tantaoui, K. A. Hua, and S. Sheu, "Interaction with broadcast video," In Proc. of ACM Multimedia, Juan-les-pins, France, December, 2002, pp. 29-38.
[76] R. Tewari, M. Dahlin, H. Vin, and J. Kay, "Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet," University of Texas at Austin February, 1998.
[77] K. Thirumalai, J.-F. Paris, and D. D. E. Long, "Tabbycat: An Inexpensive Scalable Server for Video-on-Demand," In Proc. of IEEE Int'l Conf. on Communications, Anchorage, AK, USA, May, 2003, pp. 896-900.
[78] D. A. Tran, K. A. Hua, and T. T. Do, "Scalable Media Streaming in Large Peer-to-Peer Networks," In Proc. of ACM Multimedia, Juan-les-pins, France, December, 2002, pp. 247-250.
[79] D. A. Tran, K. A. Hua, and S. Sheu, "A New Caching Architecture for Efficient Video-on-Demand Services in the Internet," In Proc. of IEEE Symposium on Applications and the Internet (SAINT 2003), Orlando, Fl, 2003, pp. 172-181.
[80] D. A. Tran, K. A. Hua, and T. T. Do, "ZigZag: An efficient peer-to-peer scheme for media streaming," In Proc. of IEEE INFOCOM, San Francisco, CA, March-April, 2003, pp. 1283-1293.

[81] D. A. Tran, K. A. Hua, and T. T. Do, "A Peer-to-Peer Architecture for Media Streaming," *IEEE Journal on Selected Areas in Communication, Special Issue on Advances in Overlay Network*, Vol. 22, No. 1, January, 2004, pp. 121-133.

[82] M. Tran, W. Putthividhya, W. Tavanapong, and J. Wong, "A Case for a Generalized Periodic Broadcast Server: Design, Analysis, and Implementation," In Proc. of IEEE Int'l Symp. on Applications and the Internet (SAINT 2004) , Tokyo, Japan, 2004, pp. 127-134.

[83] M. Tran and W. Tavanapong, "Overlay Caching Schemes for Overlay Networks," In Proc. of SPIE/ACM Multimedia Computing and Networking, San Jose, CA, USA, 2003, pp. 150-161.

[84] V. A. E. Velos, W. Meira, A. Bestavros, and S. Jin, "A Hierarchical Characterization of Live Streaming Media Workload," In Proc. of IEEE IMW'02, 2002, pp. 117-130.

[85] C. Venkatramani, O. Verscheure, P. Frossard, and K.-W. Lee, "Optimal Proxy Management for Multimedia Streaming in Content Distribution Networks," In Proc. of NOSSDAV'02, Miami, FL, USA, May, 2002, pp. 147-154.

[86] S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting," *Multimedia Systems*, vol. 4, pp. 179-208, August, 1996.

[87] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution," In Proc. of IEEE INFOCOM, vol. 3. NY, USA, June, 2002, pp. 1726-1735.

[88] K. Wu, P. S. Yu, and J. L. Wolf, "Segment-based Proxy Caching of Multimedia Streams," In Proc. of WWW'2001, Hong Kong, May, 2001, pp. 36-44.

[89] M. H. D. Xu, S. Hambrush, and b. Bhargava, "On Peer-to-Peer Media Streaming," In Proc. of IEEE Int'l Conf. on Distributed Computing and Systems, July, 2002, pp. 363-371.

[90] Z.-L. Zhang, Y. Wang, D. H. C. Du, and D. Su, "Video Staging: A Proxy-Server-Based Approach to End-to-End Video Delivery over Wide-Area Networks," *IEEE/ACM Transaction on Networking*, vol. 8, pp. 429-442, August, 2000.

[91] M. Zink, J. Schmitt, and R. Steinmetz, "Retransmission Scheduling in Layered Video Caches," In Proc. of IEEE Int'l Conf. on Multimedia Communications, Anchorage, Alaska, USA, May 2003, pp. 2474-2478.