Noname manuscript No.
(will be inserted by the editor)

A Survey on Peer-to-Peer Video Streaming Systems

Yong Liu · Yang Guo · Chao Liang

the date of receipt and acceptance should be inserted later

Abstract Video-over-IP applications have recently attracted a large number of users on the Internet. Traditional client-server based video streaming solutions incur expensive bandwidth provision cost on the server. Peer-to-Peer (P2P) networking is a new paradigm to build distributed network applications. Recently, several P2P streaming systems have been deployed to provide live and on-demand video streaming services on the Internet at low server cost. In this paper, we provide a survey on the existing P2P solutions for live and on-demand video streaming. Representative P2P streaming systems, including tree, multi-tree and mesh based systems are introduced. We describe the challenges and solutions of providing live and on-demand video streaming in P2P environment. Open research issues on P2P video streaming are also discussed.

Keywords P2P Streaming · Live · Video-on-Demand

1 Introduction

Video-over-IP applications have recently attracted a large number of users over the Internet. In year 2006, the number of video streams served increased 38.8% to 24.92 billion even without counting the user generated videos [1]. Youtube [30] alone hosted some 45 terabytes of videos and attracted 1.73 billion views by the end of August 2006. With the fast deployment of high speed residential access, such as Fiber-To-The-Home, video traffic is expected to be the dominating traffic on the Internet in near future.

The basic solution for streaming video over the Internet is the client-server service model. A client sets up a connection with a video source server and video content is streamed to the client directly from the server. One variation of client-server service model is the Content Delivery Network (CDN) based video streaming. In CDN based solution, the video source server first push video content to a set of content delivery servers placed strategically at the network edges. Instead of downloading from the video source server, a client is normally directed to a nearby content delivery server to download the video. CDN effectively shortens the users' startup delays, reduces the traffic imposed on the network, and serves more users as a whole. Youtube employs CDN to stream video to end users. The major challenge for server based video streaming solutions, though, is its scalability. A video session with good quality requires high bandwidth. With the current video compression technology, the streaming rate for a TV quality video is more than 400 kilobits-per-second. The bandwidth provision, at video source servers or in CDNs, must grow proportionally with the client population. This makes the server based video streaming solutions expensive.

Peer-to-Peer (P2P) networking has recently emerged as a new paradigm to build distributed network applications. The basic design philosophy of P2P is to encourage users to act as both clients and servers, namely as peers. In a P2P network, a peer not only downloads data from the network, but also uploads the downloaded data to other users in the network. The uploading bandwidth of end users is ef-

Yong Liu
ECE Deptartment
Polytechnic University
Brooklyn, NY 11201
E-mail: yongliu@poly.edu

Yang Guo
2 Independence Way
Thomson Lab
Princeton, NJ 08540
E-mail: Yang.Guo@thomson.net

Chao Liang
ECE Deptartment
Polytechnic University
Brooklyn, NY 11201
E-mail: cliang@photon.poly.edu

ficiently utilized to reduce the bandwidth burdens otherwise placed on the servers. P2P file sharing applications, such as [4,10], have been widely employed to quickly disseminate data files on the Internet. More recently, P2P technology has been employed to provide media streaming services. Several P2P streaming systems have been deployed to provide on-demand or live video streaming services over the Internet [6, 32,25,26]. Our recent measurement study [16] of a P2P live video streaming system shows that, in early 2006, more than $200,000$ simultaneous users watched the live broadcast of an 4-hour event at bit rates from 400 to 800 kpbs. The aggregate required bandwidth reaches 100 gigabits/sec, while Akamai reportedly has roughly 300 gigabits/sec bandwidth in its entire network at the end of year 2006.

P2P streaming systems can be broadly classified into two categories based on the overlay network structure. They are *tree-based* and *mesh-based*. The tree-based systems, such as ESM [6], have well-organized overlay structures and typically distribute video by actively pushing data from a peer to its children peers. One major drawback of tree-based streaming systems is their vulnerability to peer churn. A peer departure will temporarily disrupt video delivery to all peers in the subtree rooted at the departed peer. In a mesh-based P2P streaming system, peers are not confined to a static topology. Instead, the peering relationships are established/terminated based on the content availability and bandwidth availability on peers. A peer dynamically connects to a subset of random peers in the system. Peers periodically exchange information about their data availability. Video content is pulled by a peer from its neighbors who have already obtained the content. Since multiple neighbors are maintained at any given moment, mesh-based video streaming systems are highly robust to peer churns. However, the dynamic peering relationships make the video distribution efficiency unpredictable. Different data packets may traverse different routes to users. Consequently, users may suffer from video playback quality degradation ranging from low video bit rates, long startup delays, to frequent playback freezes.

In the rest of the article we give a survey on the existing P2P media streaming systems. The P2P live streaming systems are described first in Section 2, followed by the P2P video-on-demand systems in Section 3. You will see how different design requirements influence the system architectures. Within each section, representative systems are used as examples to show both tree-based and mesh-based system architectures. Finally, the paper is concluded with some open research problems for P2P video streaming in Section 4.

2 P2P Live Streaming

Video streaming can be classified into two categories: live and on-demand. In a live streaming session, a live video content is disseminated to all users in realtime. The video playbacks on all users are synchronized. To the contrary, video-on-demand users enjoy the flexibility of watching whatever video clips whenever they want. The playbacks of the same video clip on different users are not synchronized. In this section, we introduce several P2P live streaming systems using different overlay structures. P2P video-on-demand systems will be described in Section 3.

2.1 Tree-based Systems

In early days of the Internet, IP level multicast was proposed as an efficient way to stream audio and video to a group of users. In an IP multicast session, the video source server is connected to all users participating in the session by a multicast tree formed by IP routers in the network. Unfortunately, largely due to the router overhead of managing multicast groups and the complexity of transport control for multicast sessions, IP level multicast was never widely deployed in the Internet. Instead, the multicast function has been implemented recently at application layer. Video servers and users form an application level overlay networks to distribute video content.

2.1.1 Single-Tree Streaming

Similar to an IP multicast tree formed by routers at the network level, users participating in a video streaming session can form a tree at the application layer that is rooted at the video source server (see Fig. 1). Each user joins the tree at certain level. It receives the video from its parent peer at the level above and forward the received video to its children peers at the level below. Early examples of single-tree based streaming include Overcast [18] and ESM [6]. Figure 1 illustrates an application layer streaming tree with ten peers. There are two peers at level 1 and receiving video directly

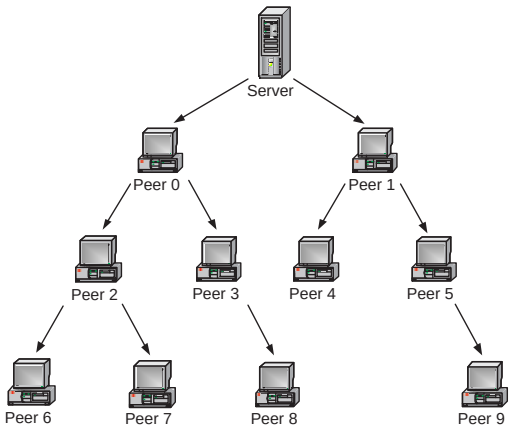


Fig. 1 Application layer multicast tree for P2P video streaming

from the server. Four peers at level 2 receive video from their parents at level 1, and three of them forward received video to four peers at the bottom level.

Given a set of peers, there are many possible ways to construct a streaming tree to connect them up. The major considerations include the depth of the tree and the fan-out of the internal nodes. Peers at lower levels of the tree receive video after peers at upper levels. To reduce the delays for peers at the bottom level, one would prefer a streaming tree with fewest levels possible. In other words, the tree topology should fan out as wide as possible at each level. However, constrained by its uploading bandwidth, a peer on an internal node can only upload video at the full rate to a limited number of children peers. The maximum fan-out degree of a peer is bounded by its uploading capacity. In fact, for the purpose of load balancing and failure resilience, the actual fan-out degree of a peer is normally set to be below its maximum degree.

Other than tree construction, another important operation for tree-based streaming is tree maintenance. Users in a P2P video streaming session can be very dynamic. A peer might leave the session at any time either gracefully or unexpectedly, e.g. machine crashes. After a peer leaves, all its descendants in the streaming tree get disconnected from the video source server and cannot receive the video any more. To minimize the disruption, the streaming tree needs to be recovered as soon as possible. Figure 2(a) illustrates a peer churn scenario when one peer close to the source server leaves. Five peers are disconnected from the video server. As shown in Figure 2(b), the streaming tree is recovered by re-assign affected peers to the server and other unaffected peers.

Tree construction and maintenance can be done in either a centralized or a distributed fashion. In a centralized solution, a central server controls the tree construction and recovery. When a peer joins the system, it contacts the central server. Based on the existing topology and the characteristics of the newly joined peer, such as its location and network access, the server decides the position of the new peer in the tree and notify it which parent peer to connect to. The central server can detect a peer departure through either a graceful sign-off signal or some type of time-out based inference. In both cases, the server recalculates the tree topology for the remaining peers and instruct them to form the new topology. For a large streaming system, the central server might become the performance bottleneck and the single point of failure. To address this, various distributed algorithms, e.g. [27], have been developed to construct and maintain streaming tree in a distributed way. However, it has been shown that tree-based streaming still cannot recovery fast enough to handle frequent peer churn.

Another major drawback of the single-tree approach is that all the leaf nodes don't contribute their uploading bandwidth. Since leaf nodes account for a large portion of peers in the system, this greatly degrades the peer bandwidth utilization efficiency.

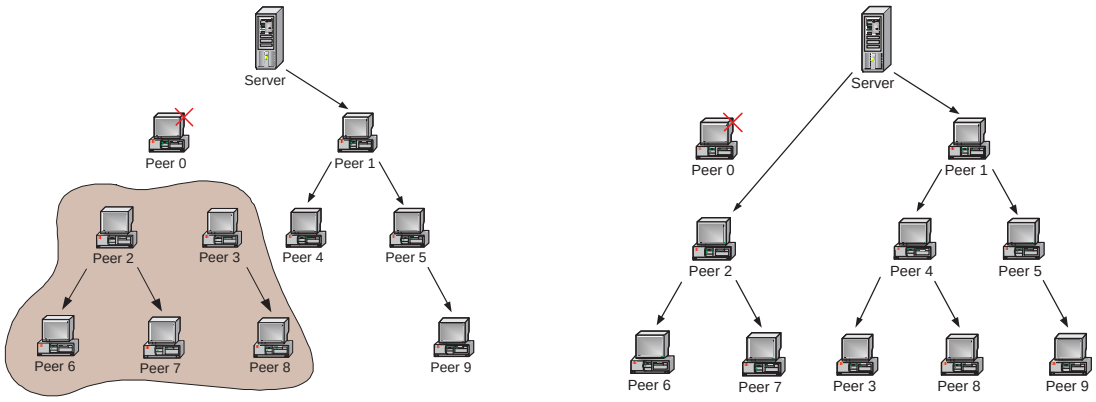
2.1.2 Multi-Tree Streaming

To address the leaf nodes problem, Multi-Tree based approaches have been proposed [5, 21]. In multi-tree streaming, the server divides the stream into multiple sub-streams. Instead of one streaming tree, multiple sub-trees are constructed, one for each sub-stream. Each peer joins all subtrees to retrieve sub-streams. Within each sub-tree, the corresponding sub-stream flows down level by level from the source server to all the leaf nodes. A peer has different positions in different sub-trees. It might be positioned on an internal node in one subtree and on a leaf node in another subtree. A peer's uploading bandwidth will be utilized to upload a sub-stream whenever it is placed on an internal node in some sub-tree. To achieve high bandwidth utilization, the number of sub-trees in which a peer is placed on an internal node can be set to be proportional to its uploading bandwidth.

In a fully balanced multi-tree streaming with m sub-streams, the node degree of each sub-tree is m. A single peer is positioned on an internal node in only one sub-tree and only uploads one sub-stream to its m children peers in that sub-tree. In each of the remaining $m-1$ sub-trees, the peer is positioned on a leaf node and downloads a sub-stream from its parent peer. Figure 3 shows an example of multi-tree streaming with 2 sub-streams and 7 peers. The server partitions the video stream into two sub-streams and push them to the left and right sub-tree respectively. Peer 0, 1 and 2 are internal nodes in the left sub-tree and leaf nodes in the right sub-tree. Similarly, peer 3, 4 and 5 are internal nodes in the right sub-tree and leaf nodes in the left sub-tree. Each peer has bandwidth of 1 and can simultaneously uploads a sub-stream of rate 0.5 to two children peers. Notice that peer 6 is a leaf node in both sub-trees and doesn't contribute to video uploading. This is because the server contributes one unit of bandwidth and only six units of peer uploading bandwidth are needed to stream to seven peers.

2.2 Mesh-based Systems

In tree-based systems, a peer has only one parent in a single streaming tree and downloads all content of the video stream (or the sub-stream for the multi-tree case) from that parent. This design introduces a single point of failure. If a peer's parent leaves, the peer, as well as its descendants, cannot get streaming feed until it connects to another parent. The management of streaming trees is challenging in face of frequent peer churns. Many recent P2P streaming systems adopt *mesh-based* streaming approach [32, 24, 28,



(a) impact of peer churn (b) tree recovery after churn

Fig. 2 Streaming tree reconstruction after a peer departure

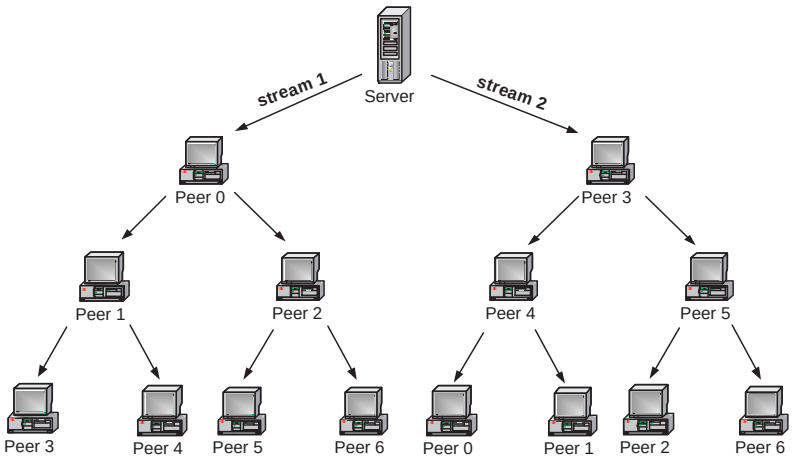


Fig. 3 Multi-Tree based streaming with two sub-streams and seven peers.

22, 31]. In a mesh-based streaming system, there is no static streaming topology. Peers establish and terminate peering relationships dynamically. At any given time, a peer maintains peering relationship with multiple neighboring peers. A peer may download/upload video from/to multiple neighbors simultaneously. If a peer's neighbor leaves, the peer can still download video content from remaining neighbors. At the same time, the peer will find new neighbors to keep a desired level of connectivity. The high peering degree in Mesh-based streaming systems makes them extremely robust against peer churn. A recent simulation study [23] suggests that mesh-based systems have superior performance than tree-based systems. In this section, we briefly describe several key design components in mesh-based systems.

2.2.1 Mesh Formation and Maintenance

Let's first look at how peers in the same video session form and maintain a mesh topology. Similar to P2P file sharing systems like BitTorrent, a mesh streaming system has a tracker to keep track of the active peers in the video session. When a peer joins the streaming session, it will contact the tracker and report its own information, such as IP address and port number, etc. Then the tracker will return a peer list that contains the information of a random subset of active peers in the session. The number of peers on a list ranges from tens to hundreds. After receiving an initial list of active peers, the peer will try to make connections to some remote peers on the list. If a connection request is accepted by a remote peer, the local peer will add the remote peer into its neighbor list. After obtaining enough neighbors, the local peer starts to exchange video content with its neighbors. Figure 4 shows the above initial setup process. To deal with frequent peer arrivals and departures, a peer constantly updates its peer list during the session. A peer can go to the tracker to ask for a fresh list of active peers. It can also find new peers by exchanging its peer list with its neighbors through the established connections. If a peer leaves the session gracefully, it will notify the tracker and its neighbors such that its information can be removed from their peer lists immediately. To

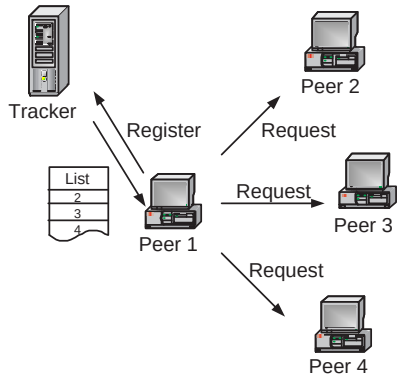


Fig. 4 Peer list retrieval from the tracker server.

handle unexpected peer departures, e.g. computer crashes, peers regularly exchange keep-alive messages. A peer will remove a remote peer's information from its list if no keep-alive message is received within a pre-configured timeout period.

A peering connection is established based on the mutual agreement between two peers at both ends. Different systems have different peering strategies, i. e., how many and which peers to connect to, when and how often to switch neighbors, etc. The peering decisions are normally made based on the following considerations:

- the workload and resource availability on both ends, such as the current number of connections, uploading and downloading bandwidth, CPU and memory usage;
- the quality of the potential connection, including the packet delay and loss characteristics on the network path between two peers;
- the content availability, i.e., how likely a remote peer will have the content needed by the local peer.

Based on those criteria, a peer not only connects to new neighbors in response to neighbor departures, but also changes neighbors voluntarily to achieve better streaming performance.

2.2.2 Data Exchange

In tree-based systems, video streams flow from the source to all peers along the streaming tree. In mesh-based systems, due to the mesh topology, the concept of video stream becomes invalid. The basic data unit in mesh-based systems is video chunk. The source server divides the video content into small media chunks, each of which contains media data for a small time interval, e.g., 0.1 second. Each chunk has a unique sequence number. A chunk with lower sequence number contains video with earlier playback time. Each chunk is then disseminated to all peers through the mesh. Since chunks may take different paths to reach a peer, they may arrive at a peer out of order. For continuous playback, a peer normally buffers received chunks in memory and put them back in order before presenting them to its video media player. Buffered chunks of one peer can be uploaded to its neighbors. Depending on the system design, a peer might keep several minutes worth of video chunks in the buffer. For live streaming, the sequence numbers of buffered chunks increases steadily as the video playback progresses.

There are two major flavors of data exchange designs in mesh-based systems: *push* and *pull*. In a mesh-push system, a peer actively pushes a received chunk to its neighbors who have not obtained the chunk yet. In tree-based system, a chunk should always be pushed from a peer to all its children peers in the streaming tree. However, there is no clearly defined parent-child relationship in mesh-based system. A peer might blindly push a chunk to a peer already having the chunk. It might also happen that two peers push the same chunk to the same peer. Peer uploading bandwidth will be wasted in redundant pushes. To address that problem, chunk push schedules need to be carefully planned between neighbors. And the schedules need to be reconstructed upon neighbor arrivals and departures.

One natural way to avoid redundant pushes is to use *pull* instead of *push*. In a mesh-pull system, peers exchange chunk availability using *buffer maps* periodically. A buffer map contains the sequence numbers of chunks currently available in a peer's buffer. After obtaining buffer maps from its neighbors, a peer can decide a chunk pull schedule that specifies from which peers to download which chunks. Then it will send requests to its neighbors to pull missing chunks. Redundant chunk transmissions can be avoided since a peer only downloads a missing chunk from only one neighbor. Frequent buffer map exchanges and pull requests do incur more signaling overhead and might introduce additional delays in chunk retrieval. In Figure 5, peer 3 generates its buffer map indicating the chunk availability in its buffer. Then it exchanges its buffer map with peer 1 and 2. Missing chunks will be requested and downloaded among all three peers.

3 P2P Video-on-Demand

Video-on-demand service (VoD) allows users to watch any point of video at any time. Compared with live streaming, VoD offers more flexibility and convenience to users and truly realizes the goal of *watch whatever you want whenever you want*. VoD has been identified as the key feature to attract consumers to IPTV service.

In VoD service, although a large number of users may be watching the same video, they are asynchronous to each other and different users are watching different portions of the same video at any given moment. Tree-based P2P system is originally designed to function as IP multicast at the application layer without underlying network layer's support.

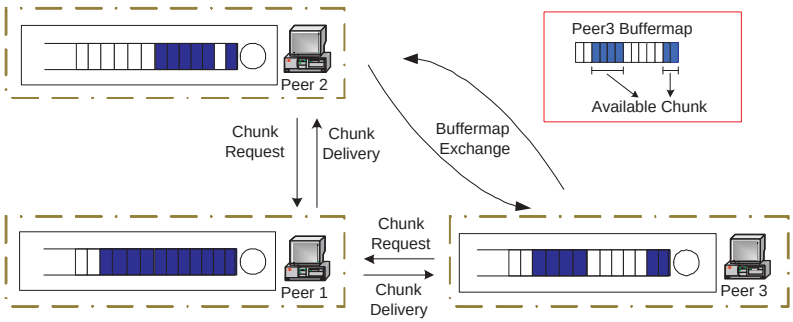


Fig. 5 Buffer map exchange and data pull among peers.

The users using tree-based overlay is synchronized and receive the content in the order the server sends it out. This is fundamentally different from the requirement imposed by VoD service. How to accommodate asynchronous users using tree-based P2P system is a challenging design issue.

Mesh-based P2P system is first introduced to distribute large files and then successfully applied to live streaming. Typically a large file is divided into many small blocks. The system throughput and the rate at which the content can be distributed to users heavily depend on the diversity of content blocks available at different peers. The order at which the blocks are received is different from peer to peer and is very random. The challenges to offer VoD using mesh-based P2P network is two folds. At the peer-level, the content blocks have to be received before their playback time. Ideally, the content blocks should be downloaded in the same order as in the source file. At the system level, the content sharing has to be enabled among asynchronous peers and the overall system throughput has to be high even with the per-peer downloading constraint. Supporting VoD using mesh-based P2P is again not straight-forward.

In the following, we present three representative solutions that have been developed in the past to support VoD using tree-based and mesh-based P2P system. As described in the previous section, tree-based and mesh-bashed P2P systems have their own pros and cons. Here we focus on how to adapt these approaches to providing VoD service.

3.1 Tree-based P2P VoD

Inspired by the patching scheme [17,11] proposed to support VoD service using native IP multicast, the authors in [14] designed a scheme that uses tree-base P2P system to support asynchronous users in VoD service.

Users are grouped into sessions based on their arrival time. A threshold, T, is pre-defined. The users that arrive close in time and within the threshold constitute a session. Together with the server, users belonging to the same session form an application-level multicast tree, denoted as the base tree. The server streams the entire video over the base tree as in tree-based P2P live streaming. This complete video stream is denoted as the base stream. When a new client joins the session, it joins the base tree and retrieves the base stream from it. Meanwhile, the new client must obtain a *patch* - the initial portion of the video that it has missed (from the start of the session to the time it joined the base tree). The patch is available at the server as well as other users who have already cached the patch. Users behave like peers in the P2P network, and provide the following two functions:

- Base Stream Forwarding: Users participate in the tree-based overlay and forwards the received base stream to its children. The base stream is shared among all users in the tree.
- Patch Serving: Users cache the initial part of the video and serve the patch to latecomers.

Fig. 6 illustrates a snapshot of the above solution when a new user arrives at time 40. It shows two sessions, session 3 and session 4, starting at time 20.0 and 31.0, respectively, with the threshold equal to 10. Each user is marked with its arrival time to the system. A solid line with an arrow is used to represent the parent-child relationship in the base tree; and a dashed line with an arrow is used to represent the patch server-client relationship. The server and the clients in a session form an application-level multicast tree to deliver the base stream. At time 40, all clients in session 3 have finished the patch retrieval; while three clients in session 4 are still in the process of receiving the patch stream. Note that users belonging to different sessions do not interact with each other.

Note that users are synchronous in the base tree. The asynchronous requirement of VoD is addressed using patching. In the following, we describe cache-and-relay P2P VoD. It again employs tree-based approach, however, the asynchronous issue is solved by the content caching at users.

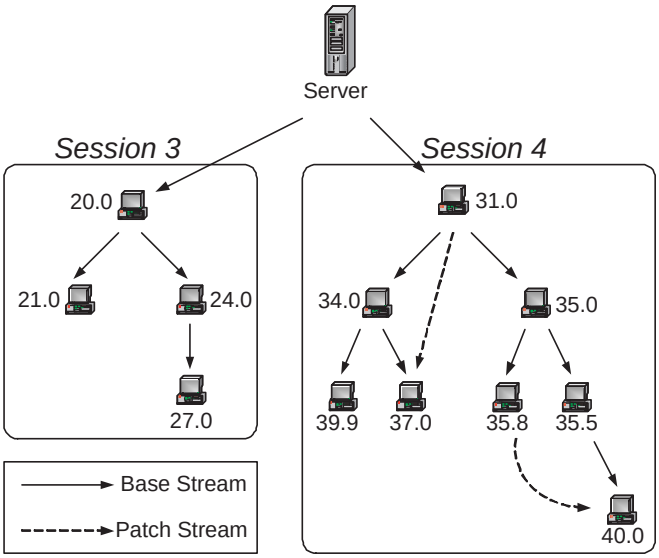


Fig. 6 A snapshot of the scheme at time 40. Users belonging to the same session form an application-level multicast tree together with the server. Users in session 3 have finished patch retrieval; while 3 clients in session 4 are still receiving the patch stream from their parent patch servers.

3.2 Cache-and-Relay P2P VoD

To efficiently utilize memory, a streaming server caches a moving window of video content in the memory so as to serve a batch of clients whose viewing point falling into the caching window. This is so-called interval caching technique [20,8]. Cache-and-relay P2P VoD applies the interval caching idea to solve the asynchronous issue in tree-based P2P VoD. A peer in a cache-and-relay P2P VoD system buffers a moving window of video content around the point where they are watching. It serves other users whose viewing point is within the moving window by continuously forwarding the stream. Although a P2P tree is formed among peers, their playback points are different and the synchronization issues is successfully addressed.

Fig. 7 illustrates a simple example of cache-and-relay P2P VoD system. Here users are assumed to watch the video from the beginning and cache 10 minutes worth of video data. User *A* arrived first at time 1. Since there is no other users in the system, it retrieves the video from the server directly. Later on, user *B* and *C* arrived at time 3 and 8, respectively. Both discover that user *A*'s buffer still covers the beginning of the video. They manage to ask user *A* to forward the stream from the very beginning. When user *F* joined the system at time 50, however, the moving windows of early arrivals have passed the video beginning. User *F* is forced to retrieve the video from the server directly. As time goes, latercomers are able to obtain the video from user *F* and its descendants.

In this example, the users form two clusters, cluster 1 and cluster 2. A cluster represents a set of users that are able to share a single stream out of the server. A tree is established among the users of the same cluster. For instance, user *A, B, C* and three other users form the first cluster. The video stream is cached and relayed along the path from the source all the way to the users placed at the bottom of the tree. A parent user's moving buffer always covers the child peer's playback point.

Interestingly, a cluster in cache-and-relay P2P VoD evolves over time. For instance, user A was a member of the cluster 1. However it left the cluster after finishing the playback and forwarding the video to B and C (see Fig. 7(b)). From now on, no video out of the server is needed for cluster 1 users. In the extreme case, if users arrive close in time, server only needs to stream the video to the first user. The followers can form a chain and obtain the service from early arrivals. Cache-and-Relay approach proves to be a very scalable solution.

In both tree-based and cache-and-relay P2P VoD, the construction of overlay tree and the handling of peer churn remain to be key issues. For cache-and-relay based P2P VoD, it also imposes extra constraints where a user only has limited number of users who can be its parents, i.e, users can be a candidate parent only if its caching window cover the viewing point of child user. A directory service is also required to facilitate locating candidate parents. The works in [19,15,7,2,12] addressed various issues arisen in designing cache-and-relay based P2P VoD service. Jin et al. [19] derived bounds on the network cost of cache-and-relay approach. Guo et al. [15] studied the workload posed on the server and showed that the system scales even if the peers behave no-cooperatively. In [12], the authors further developed an application-level multicast based directory service tailored for cache-and-relay P2P VoD. Cui et. al [7] ana-

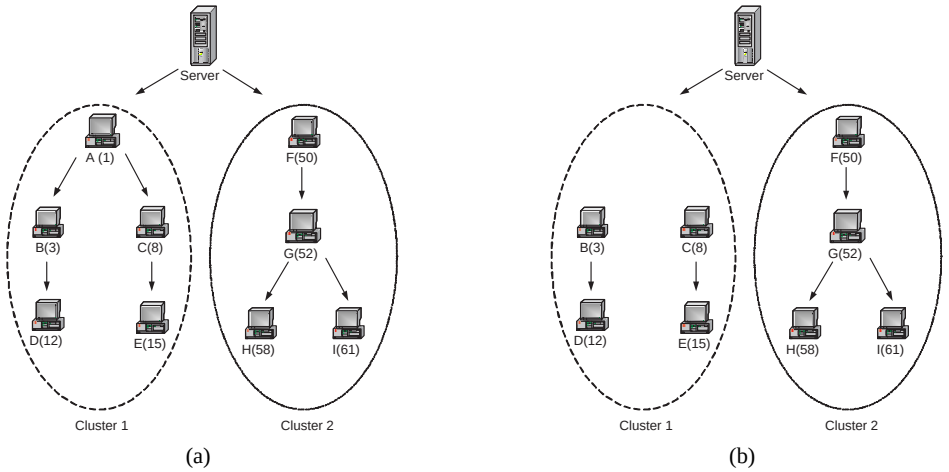


Fig. 7 DirectStream system. (**a**) DirectStream system with two clusters — one headed by client A and the other headed by client F. (**b**) DirectStream system after the departure of client A. No service from the server is required from now on.

lyzed the server bandwidth requirement and network-wide link bandwidth requirement under both sequential and non-sequential stream access patterns. Their work shows that cache-and-relay scheme defeats IP multicast-based VoD scheme in terms of both server bandwidth consumption and network bandwidth consumption. Finally, Sharma et al. [2] introduced the prefetching technique into cache-and-relay to overcome the peer churn, and examined its impact on the server bandwidth requirement.

3.3 Mesh-based P2P VoD

Mesh-based P2P file sharing network achieves fast file downloading by swarming. A file is divided into small size data blocks. The server (typically called seed in the mesh-based P2P network context) disperses the data blocks to different users. The users download from its neighboring peers the blocks that they currently don't have . To fully utilize users upload bandwidth and hence achieve highest downloading throughput possible, the data blocks at different users are better-off to be different from each other so that there is always something to exchange. This is so-called the diversity requirement in mesh-based P2P system.

The diversity improves the systems overall throughput. However, the effective rate at which users can playback a video file may not be good. This is obvious since the data blocks are retrieved in a fairly random order while the video blocks have to be played in sequential order. Moreover, due to the asynchronous nature of VoD service, the users are interested in different parts of content at any given moment. The availability of different content blocks is also skewed by users behavior. Therefore the challenge of designing a mesh-based P2P VoD scheme rests on the right balance between the overall system efficiency and the conformation to the sequential playback requirement for asynchronous users.

BiToS [29] probably is the first attempt to design a mesh-based P2P VoD service system and we use it as an example of mesh-bashed P2P VoD service here. A peer in BiToS has three components as shown in Fig. 8. The received buffer stores all the data blocks that have been received so far. High priority set contains the video blocks that close to their playback time yet have not been downloaded. The remaining piece set contains the blocks that have not been downloaded. The scheme uses a selection process to decide which block to download. A block in high priority set is downloaded with probability p while the one in remaining pieces set is downloaded with probability $1-p$. By setting the value of p greater than 0.5, the blocks in high priority set are favored to be downloaded earlier than the ones in the remaining pieces set. Intuitively, the larger value of p offers better chance for the blocks to arrive before their playback time, while smaller value of p increases the diversity of blocks and hopefully leads to better overall system efficiency.

Although the scheduling at individual peers in mesh-based P2P VoD may look similar to the one in mesh-based P2P live streaming, the difference lies in the fact that in VoD users are asynchronous and watching different part of video. In P2P live streaming, peers are interested in the similar part of video. Whatever data units downloaded at a peer is also useful to other peers that have not retrieved those data units. In P2P VoD, if the video is downloaded in the order of playback, a newly arrived user can make little contribution because it doesn't have the content other earlier arrived users are looking for. Meanwhile, many earlier arrived users can serve the content to the new arrival since they have watch the beginning part of the video. In contrast, as time goes on, a peer caches more and more data and can serve more peers.

Fig. 8 BiToS peer structure.

However, the number of peers that can upload content to this peer decreases since some peers arrived before this peer may have finished watching and left the system. How to optimally allocate the resources across different parts of the video and how to manage the overlay topology are important design questions. The works [13, 3] divide the video into segments with each segment containing a set of blocks. The segments close to the playback point is given high priority to download. [3] also employs network coding to improve the resource utilization efficiency. The works in [9, 13] introduce the source server to help out in case the video content is not available when the playback time is imminent.

4 Conclusions and Open Issues

In this paper, we conducted a survey on the existing P2P video streaming technology. We described several key P2P streaming designs, including system topologies, peering connections and data scheduling, that address various challenges in providing large scale live and on-demand video streaming services on top of the best-effort Internet. Current deployments on the Internet demonstrate that P2P streaming systems are capable of streaming video to a large user population at low server cost and with minimal dedicated infrastructure. However, there are several fundamental limitations of existing P2P video streaming solutions.

First of all, the user Quality of Experience in current P2P streaming systems are still not comparable to the traditional TV services provided by cable and satellite broadcasting companies. Specifically, P2P streaming users normally experience much longer channel start-up and channel delays. Video playback starts tens of seconds after a user selects a channel. There are also large playback lags among peers. Some peers watch frames in a channel minutes behind other peers. Due to the limited peer uploading capacity, most P2P streaming systems only support video rate up to 400kbps. Consequently, users only receive low resolution videos. In addition, the video streaming quality is poor and unstable when the number of peers watching the same program is small. This makes it challenging to serve long-tailed unpopular contents in P2P streaming systems. Those issues are interesting and challenging research problems that need to be addressed to make P2P video streaming services a real competitor for traditional broadcast TV services.

Secondly, the increasing popularity of P2P streaming has become a serious concern for ISPs. The huge user base and high traffic volume of P2P streaming systems pose a big challenge on ISPs' network capacities. Most current P2P streaming designs are not *ISP-Friendly*. The peering connections and data exchanges among peers are mostly driven by content availabilities. After peers obtain some video data from the source server, they randomly connect to multiple peers, local and remote, and exchange data between different networks. Unregulated P2P video exchanges significantly increase the traffic volume on links within and between ISPs. As a result, the video content distribution cost is essentially shifted to ISPs without any profit for them. How ISPs should manage and regulate the ever increasing P2P video streaming traffic deserves further investigation to maintain the stability of their network infrastructures.

Lastly, playing the dual role of network service provider and content service provider, several ISPs have started to provide IPTV services by deploying IP multicast and video proxy servers in their private networks. P2P streaming has been proved to be a scalable streaming solution with low infrastructure requirement. It will be beneficial for ISPs to integrate P2P technology into their IPTV systems to significantly reduce their server and network infrastructure cost. Many interesting research problems need to be addressed to develop an integrated IPTV solution for content providers, network providers and IPTV users.

References

1. Accustream iMedia Research Homepage. `http://www.accustreamresearch.com`.
2. Abhishek Sharma, A. B., and Matta, I. dpam: A distributed prefetching protocol for scalable asynchronous multicast in p2p systems. In *Proceedings of IEEE INFOCOM* (Mar. 2005).

3. ANNAPUREDDY, S., GUHA, S., GKANTSIDIS, C., GUNAWARDENA, D., AND RODRIGUEZ, P. Is high-quality vod feasible using p2p swarming? In *the 16th International World Wide Web Conference (WWW2007)* (May 2007).
4. BT. Bittorent Homepage. `http://www.bittorrent.com`.
5. CASTRO, M., DRUSCHEL, P., KERMARREC, A.-M., NANDI, A., ROWSTRON, A., AND SINGH, A. SplitStream: High-bandwidth multicast in cooperative environments. In *Proceedings of ACM SOSP* (2003).
6. CHU, Y.-H., G.RAO, S., AND ZHANG, H. A case for end system multicast. In *Proceedings of ACM SIGMETRICS* (2000).
7. CUI, Y., LI, B., AND NAHRSTEDT, K. ostream: Asynchronous streaming multicast in application-layer overlay networks. In *IEEE Journal on Selected Areas in Communications* (Jan. 2004).
8. DAN, A., AND SITARAM, D. A generalized interval caching policy for mixed interactive and lo ng video enviroments. In *SPIE Multimedia Computing and Networking Conference* (Jan. 1996).
9. DANA, C., LI, D., HARRISON, D., AND CHUAH, C. Bass: Bittorrent assisted streaming system for video-on-demand. In *International workshop on multimedia signal processing (MMsP)* (2005).
10. EMULE. Emule Homepage. `http://www.emule-project.net`.
11. GAO, L., AND TOWSLEY, D. Threshold-based multicast for continuous media delivery. In *IEEE Transactions on Multimedia* (Dec. 2001).
12. GUO, Y., , SUH, K., KUROSE, J., AND TOWSLEY, D. Directstream: A directory-based peer-to-peer video streaming service. Tech. rep., UMass CMPSCI Technical Report TR 07-30, 2007.
13. GUO, Y., MATHUR, S., RAMASWAMY, K., YU, S., AND PATEL, B. Ponder: Providing commercial-quality video-on-demand service using peer-to-peer network. In *Technical repot, Corporate Research, Thomson Inc.* (July 2006).
14. GUO, Y., SUH, K., KUROSE, J., AND TOWSLEY, D. P2cast: Peer-to-peer patching scheme for vod service. In *Proceedings of the 12th World Wide Web Conference (WWW-03)* (May 2003).
15. GUO, Y., SUH, K., KUROSE, J., AND TOWSLEY, D. A peer-to-peer on-demand streaming service and its performance evaluation. In *Proceedings of 2003 IEEE International Conference on Multimedia & Expo (ICME 2003)* (July 2003).
16. HEI, X., LIANG, C., LIANG, J., LIU, Y., AND ROSS, K. A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia* (November 2007).
17. HUA, K., CAI, Y., AND SHEU, S. Patching: A multicast technique for true video-on-demand services. In *Proc. of ACM Multimedia* (Sept. 1998).
18. JANNOTTI, J., GIFFORD, D. K., JOHNSON, K. L., KAASHOEK, M. F., AND O'TOOLE, JR., J. W. Overcast: Reliable multicasting with an overlay network. In *Proceedings of Operating Systems Design and Implementation* (2000), pp. 197–212.
19. JIN, S., AND BESTAVROS, A. Cache-and-relay streaming media delivery for asynchronous clients. In *International Workshop on Networked Group Communication* (Oct. 2002).
20. KAMATH, M., RAMAMRITHAM, K., AND TOWSLEY, D. Continuous media sharing in multimedia database systems. In *Proc. of 4th International Conference on Database Systems for Advanced Applications (DASFAA'95)* (Apr. 1995).
21. KOSTIC, D., RODRIGUEZ, A., ALBRECHT, J., AND VAHDAT, A. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proceedings of ACM Symposium on Operating Systems Principles* (2003).
22. MAGHAREI, N., AND REJAIE, R. Prime: Peer-to-peer receiver-driven mesh-based streaming. In *Proceedings of IEEE INFOCOM* (2007).
23. MAGHAREI, N., REJAIE, R., AND GUO, Y. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *Proceedings of IEEE INFOCOM* (2007).
24. PAI, V., KUMAR, K., TAMILMANI, K., SAMBAMURTHY, V., AND MOHR, A. Chainsaw: Eliminating trees from overlay multicast. In *The Fourth International Workshop on Peer-to-Peer Systems* (2005).
25. PPLIVE. PPLive Homepage. `http://www.pplive.com`.
26. PPSTREAM. PPStream Homepage. `http://www.ppstream.com`.
27. TRAN, D. A., HUA, K., AND DO, T. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In *Proceedings of IEEE INFOCOM* (2003).
28. VENKATARAMAN, J. C. V., AND FRANCIS, P. Multi-tree unstructured peer-to-peer multicast. In *Proceedings of 5th International Workshop on Peer-to-Peer Systems* (2006).
29. VLAVIANOS, A., ILIOFOTOU, M., AND S, M. F. Bitos: Enhancing bittorrent for supporting streaming applicati ons. In *9th IEEE Global Internet Symposium 2006* (Apr. 2006).
30. YOUTUBE. Youtube Homepage. `http://www.youtube.com`.
31. ZHANG, M., ZHAO, L., TANG, J. L. Y., AND YANG, S. A peer-to-peer network for streaming multicast through the internet. In *Proceedings of ACM Multimedia* (2005).
32. ZHANG, X., LIU, J., LI, B., AND YUM, T.-S. P. DONet/CoolStreaming: A data-driven overlay network for live media streaming. In *Proceedings of IEEE INFOCOM* (2005).