IPTV over P2P Streaming Networks: the Mesh-pull Approach

Xiaojun Hei†, Yong Liu‡ and Keith W. Ross†
†Department of Computer and Information Science
‡Department of Electrical and Computer Engineering
Polytechnic University, Brooklyn, NY, USA 11201

***Abstract*—An emerging Internet application, IPTV, would revolutionize the entertainment and media industries; however, IPTV has the potentials to overwhelm the Internet backbone and access networks with traffic. To date, IPTV over P2P streaming networks have advanced significantly in two major approaches: tree-push vs. mesh-pull. In particular, the mesh-pull streaming approach has achieved a number of successful commercial deployments. In this article, we examine the current progress on the research and development of mesh-pull P2P streaming systems. We provide an overview of the general mesh-pull streaming architecture, review various challenges, design issues, and interesting research problems in this approach. We discuss the construction cost to provide an IPTV service with service guarantees. We outline a measurement technique for monitoring the video playback quality of mesh-pull streaming systems. We emphasize that the future P2P IPTV systems should be designed towards users' Quality-of-Experience (QoE). We also identify a few other important issues for IPTV over P2P streaming networks, including traffic pressure on ISPs, various security concerns and the necessity of re-examination of the most appropriate P2P architecture. Insights obtained in this study will be valuable for the development and deployment of future P2P IPTV systems.**

***Index Terms*—IPTV, Peer-to-peer streaming**

I. Introduction

With the widespread adoption of broadband residential access and the advances in video compression technologies, IPTV may be the next disruptive Internet application. Considering the scenarios that hundreds of millions of users watch video programs online at bit rate 500 kbps or more, IPTV would revolutionize the entertainment and media industries; however, IPTV has the potentials to overwhelm the Internet backbone and access networks with traffic. Given this possible tidal wave of new Internet traffic, an in-depth understanding of the IPTV delivery mechanisms, particularly for the delivery architectures that hold the greatest promise for broad deployment in the near future.

P2P streaming networks do not rely on a dedicated delivery infrastructure and hence offer the possibility of rapid deployment at low cost. The upload capacity of peers can be utilized for video transmission so as to reduce the server load dramatically. Therefore, P2P streaming networks have appeared to be the most promising driving wheel for the IPTV deployment. Content owners are often not equipped with a strong content delivery infrastructure; the low cost incurred in the P2P streaming paradigm is particularly appealing to these content owners. Nevertheless, the huge video traffic volume generated by P2P applications has been raising high traffic load on the network infrastructures of telecommunication operators.

There exist two major design issues for constructing a P2P streaming network: (i) how to form an overlay topology between peers; (ii) how to deliver video content efficiently. The current approaches can be classified into two categories: (i) peers form a tree-shaped overlay and video content is pushed from the origin server to peers, namely, the *tree-push* approach; (ii) peers form a mesh-shaped overlay and they pull video from each other for content delivery, namely, the *mesh-pull* approach. Over years, many tree-push systems have been proposed and evaluated in academia and achieved some successes. However, they have never taken off commercially. Nevertheless, mesh-pull IPTV systems have enjoyed a number of successful deployments to date, such as CoolStreaming [1], PPLive (www.pplive.com) and others ([2], [3], [4]). The major advantages of mesh-pull systems include the simple design principle and inherent robustness particularly desirable for highly dynamic, high-churn P2P environment. A well-written tutorial on the P2P video streaming over the Internet can be found in [5]. Our study in this article dissects the general architecture of the current most popular mesh-pull P2P streaming systems.

The aim of this article is to explore the general design space and challenges of mesh-pull systems. We first provide an overview of the general mesh-pull streaming architecture in Section II. The operation of the mesh-pull system rely on the bandwidth contribution from each peer. Peers may have different upload capacity due to different types of access links. In section III, we discuss the implication of the heterogeneity in peer uploading capacities on the system scalability. User experiences play an important role in a successful IPTV application. We then review the service quality issues and introduce a quality monitoring methodology for mesh-pull systems in Section IV. Mesh-pull streaming systems employ both TCP and UDP as the underneath transport control protocol. The congestion control problem is re-visited in Section V. Security is one vulnerable aspect in most current mesh-pull systems. In section VI, we outline the potential peer poison and video pollution attacks in these systems. Different P2P streaming architectures have different advantages and drawbacks on video playback quality perceived by end-users. Pure mesh-pull architectures may be difficult to meet the stringent delay requirement of future IPTV services. We briefly discuss in

Section VII a hybrid tree-mesh architecture for the next generation P2P IPTV systems. Finally, we conclude the article with highlighting the design challenges for future mesh-pull streaming systems.

II. Mesh-pull P2P Streaming Architecture

A number of mesh-pull streaming systems have been deployed to date. However, most mesh-pull streaming systems provide little information about their proprietary technologies. Through our measurement studies and protocol analysis on a few mesh-pull streaming systems [2], [6], we have gained significant insights into the protocols and streaming mechanisms of mesh-pull streaming systems. In spite of different features and implementation details, they share a common generic architecture, which we describe in this section.

A. System Overview

As shown in Fig. 1, mesh-pull P2P architectures have the following characteristics. A video is divided into media chunks and is made available from an origin server for broadcast. All the video information is accessible for users at the channel server. A host, interested in viewing the video, requests from the channel server for the available video streams (Step 1 in Fig. 1). The tracker server maintains the list of the hosts who are interested in watching the same video. After a host selects its interested video, it retrieves a list of hosts currently watching the same video (Step 2 in Fig. 1). The host then establishes partner relationships (TCP/UDP connections) with a subset of hosts on the list (Step 3 in Fig. 1). These peers help each other and deliver video traffic cooperatively. The host may also establish a partner relationship with the origin server. Each host viewing the video caches and shares chunks with other hosts viewing the same video. In particular, each host receives *Buffer Maps (BMs)* from its current partners. A buffer map from a remote partner indicates the chunks that are available on that partner. Using a chunk scheduling algorithm, each host requests from its partners the chunks that it will need in the near future. Each host continually seeks new partners from which it can download chunks.

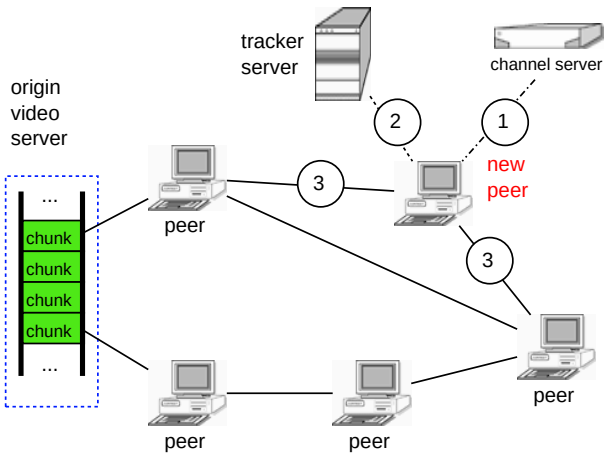

Fig. 1. Mesh-pull P2P live streaming architecture.

B. Software Components

Fig. 2 shows the software components of a peer in a mesh-pull system. The peer software includes a P2P streaming engine and a media player. The streaming engine has the job of (i) retrieving chunks from partner peers and (possibly) from the origin server; (ii) storing the retrieved chunks in a cache; (iii) sharing media chunks stored in its cache with its partners; (iv) sending a copy (of the data) of each chunk, which it receives, to the media player. As shown at the bottom of Fig. 2, the local peer sends a buffer map to each of its partner peers. A partner peer, having learned from the buffer map what chunks which the local peer has, issues requests for specific chunks. The local peer then sends the requested chunks to that partner peer.

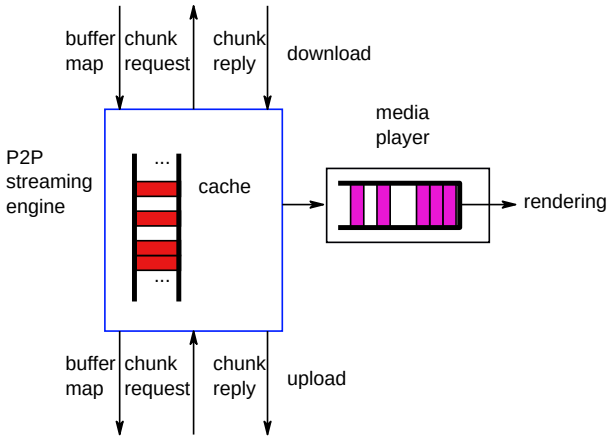

Fig. 2. A peer includes P2P streaming engine and media player. The streaming engine trades chunks with partner peers.

1) P2P Streaming Engine: At any given instant, the P2P streaming engine caches up to a few minutes worth of chunks within a sliding window. Some of these chunks may be chunks that have been recently played; the remaining chunks are chunks scheduled to be played in the next few minutes. Peers download chunks from each other. To this end, peers send to each other buffer map messages; a buffer map message indicates which chunks a peer currently has buffered and can share. A buffer map message includes the offset (the ID of the first chunk), the width of the buffer map, and a string of zeroes and ones indicating which chunks are available (starting with the chunk designated by the offset). A unique channel ID is also carried in each buffer map message. Fig. 3 illustrates the structure of a buffer map.

A peer can request a buffer map from any of its current partner peers. After peer A receives a buffer map from peer B, peer A can request one or more chunks that peer B has advertised in its buffer maps. A peer may download chunks from tens of other peers simultaneously. The streaming engine continually searches for new partners from which it can download chunks. Different mesh-pull systems may differ significantly with their peer selection and chunk scheduling algorithms. The selection and scheduling algorithms used by CoolStreaming are documented in [1]. Peers can also download chunks from the origin server. The chunks are typically

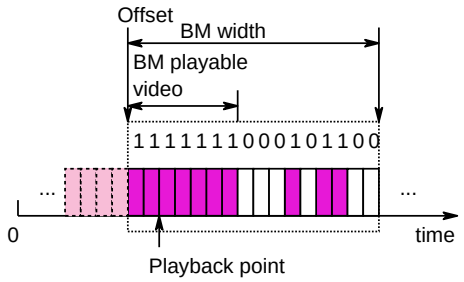

Fig. 3. A peer's buffer map, which indicates the chunks it currently has cached.

sent over TCP connections, although recently in more mesh-pull systems, video chunks are also transferred using UDP.

2) Media Player: When the client application is started, the media player is launched and the URL of the video stream is provided to the media player. From the client's perspective, the server of the media content is the P2P streaming engine (which is in the same host as the media player). Once the media player is initialized, it sends (typically) an HTTP request to the P2P streaming engine. After having received the request, the P2P streaming engine assembles its chunks and header information into a media file and delivers the file to the media player. Because new chunks continually arrive to the streaming engine, the streaming engine continually adds data to the file. Because some chunks may not arrive before the playback deadline, there may be "gaps" in the received media file. When the media player begins to receive the video from the streaming engine, it buffers the video before playback. When it has buffered a sufficient amount of continuous video content, it begins to render the video.

C. Mesh-pull streaming: BitTorrent revisited?

Bearing strong similarities to BitTorrent, mesh-pull streaming deviates significantly from BitTorrent in various aspects. BitTorrent by itself is not a feasible video streaming architecture, since it does not account for the real-time needs of IPTV. In mesh-pull streaming, each video chunk has corresponding playback deadline. Hence, video chunk scheduling is an indispensable component for assisting a timely video delivery.

Due to the stringent playback deadline of video chunks, fair resource sharing has not been carefully addressed in the current mesh-pull systems, in that, there have been no reciprocity mechanisms, such as Tit-for-Tat equipped in BitTorrent, deployed in the current mesh-pull systems to encourage sharing between peers.

BitTorrent is targeted at the group communication with medium size (< 1000); hence, peers retrieve peer neighbor information directly from the tracker server. However, a large-scale live streaming broadcast can easily attract tens of thousands of users. Hence, gossip-like peer search algorithms have been implemented in various mesh-pull systems to support large-scale group communication. However, the deployment of gossip algorithms incur various implications, i.e., delay may occur in searching peers; tracker servers may only handle part of the peers in the system and hence lose the global view and the control of the network, and so on.

III. System Scalability: Construction Cost

In mesh-pull streaming systems, participating peers are very heterogeneous, particularly in terms of the amount of upload bandwidth they contribute [2]. In addition, peers may randomly join the system, watch the video for a random period of time, and then leave the system. These two factors, peer heterogeneity and churn, bring forth the major challenges in provisioning the P2P IPTV services so that all participating peers can continuously playback the video (without freezing or skipping) with a small playback delay.

The current peers participating mesh-pull streaming systems are usually broadband residential peers with DSL and cable access, and institutional peers with high-bandwidth Ethernet access. Those residential peers, or ordinary peers, typically have upload capacity of 500 kbps or less; the institutional peers, or super peers, often have upload capacity in excess of a few Mbps. The service level of P2P streaming systems is largely determined by the resources contributed by all the peers in the system, i.e., upload bandwidth. These two types of peers, ordinary and super peers, have different upload capacities. If the upload capacity of ordinary peers cannot sustain the video playback rate, super peers have to contribute additional upload capacities to help those ordinary peers. As demonstrated in [7], there exists a critical value of the ratio between ordinary peers and super peers for sustaining a satisfied service level of P2P streaming systems. Roughly speaking, when this ratio exceeds this critical value, the system performs well; otherwise, the system performs poorly.

Most mesh-pull streaming systems to date only deploy limited delivery infrastructure, i.e., channel list servers, tracker servers, etc.. The service quality of peers varies significantly at different time and locations. IPTV service providers may invest on additional streaming infrastructure, either using servers that they operate or that are provided by Content Distribution Networks (CDN). To be cost-effective, it is crucial that service providers detect quickly when service quality degrades, so that they can add, retroactively and dynamically, additional uploading capacity. When service quality is satisfactory, the infrastructure capacity can be released for other purposes.

IV. Video playback: QoS towards QoE

The users' viewing experience in IPTV is crucial for a successful service deployment. If users experience frequent freezes in the video playback, significant delays for startup after switching channels, or significant time lags among users for the same video frame, then the users may abandon the service. Important IPTV quality metrics include initial startup latency, video switching latency, display lags among users, video playback continuity, etc..

A. Quality metrics

As theoretically demonstrated in [7], appropriate buffering can significantly improve video streaming quality. However, too much buffering in mesh-pull systems may lead to an unacceptable delay performance for an IPTV service. *Start-up delay* is the time interval from when one video is selected by a user until actual playback starts on his/her screen. For

IPTV applications in the best-effort Internet, start-up buffering has always been a useful mechanism to deal with the rate variations of IPTV sessions. IPTV applications additionally have to deal with peer churn, increasing the need for startup buffering and delay. While short start-up delay is desirable, certain amount of start-up delay is necessary for continuous playback. End-users may also switch to watch another video from the current one. Before users are able to watch the new video, the buffering in mesh-pull systems often incur *video switching delay*. These delays are, of course, significantly longer than what are provided by traditional television. Hence, the state-of-the-art mesh-pull P2P streaming technology does not provide users with the same channel-surfing experience as the traditional television.

Another unfortunate characteristic of a mesh-pull P2P streaming system is the possibility of *playback time lags* among peers due to the deployment of the buffering mechanisms. Specifically, some peers watch frames in a video minutes behind other peers. Thus, for the example of live broadcast of a soccer game, some peers might see a goal a few minutes after other peers. Additionally, peers with large playback lags will not upload useful chunks to peers with smaller lags, decreasing the aggregate uploading capacity of the system.

In addition, because of the real-time nature of IPTV, each media chunk has a playback deadline (which can be different from one peer to another by a few minutes). When a chunk does not arrive before its playback deadline, the peer has two options: it can *freeze* the playback with the most recently displayed video frame and wait for the missing chunk to arrive; or it can *skip* the playback of the frames in the chunk and advance the deadlines for the subsequent chunks accordingly. The freeze is a natural consequence of the media playback when there are no video chunks in the buffer of the player. If there are still chunks available in the buffer of the player, the player continues playback even though those chunks might not be continuous; in this case, video playback skipping occurs. In many P2P live streaming systems, when the playback freezes for an extended period of time, the engine terminates the connection with the player and reinializes the entire streaming process; we refer to this impairment as *rebooting*.

B. Quality inference using buffer map

Various parties are interested in monitoring service quality of IPTV applications. Service providers would like to detect when service quality degrades, so that they can add additional uploading capacity for maintaining a satisfied service level. This information could be provided to users as an aid in selecting P2P video providers. It could also be provided to advertisers who wish to advertise in IPTV systems.

Video quality is conventionally characterized by video distortion; the quality measurement normally involves the video rate measurement from which video distortion is estimated utilizing empirical rate-distortion curves or various proposed rate-distortion models. However, this approach usually varies among the video sequences, and is not suitable for video quality measurement over the Internet because this approach typically requires extensive direct traffic measurements and hence incurs high overhead and cost.

As we describe earlier in Section II, peers in P2P mesh-pull streaming systems advertise buffer maps to each other for exchanging the availability of the video chunks that they currently have cached. The information provided in these buffer maps correlates with that peer's playback quality and startup latency. Given this correlation, buffer maps can be exploited to measure network-wide quality [6]. In this approach, instead of monitoring video traffic between peers, we capture the exchange of peers' buffer maps, which summarize the availability of video data at the various peers. These buffer maps are of very small sizes. Therefore, this approach generates significantly less measurement traffic in the network.

As shown in Fig. 3, the offset value in a buffer map indicates the playback position of a peer. The string of zeroes and ones embedded in buffer maps shows which chunks a peer currently has buffered and can share. When a peer joins the network and has no chunks in its engine cache, it sends buffer maps with a particular playback offset, i.e., equal to zero. After a peer obtains chunks from other peers, the offset is set to a value of the playback starting position. The streaming engine will start the player when the size of the cached playable video size exceeds a specified pre-buffering video threshold. The consecutive buffer maps provide snapshots of the video chunks in the cache. Therefore, we can infer the start-up delay of the playback of a peer given the sequence of the buffer maps of this peer. In addition, by tracing the switching of the channel IDs carried in buffer maps, we are able to infer the channel switching delay of one peer. In tracking user's playback continuity, the playback freezing events are inferred when the BM playable video, determined from the buffer maps, remains at a low level of sizes over a period of time. Under normal circumstances, the buffer map offset should increase at the video playback rate; hence, by tracking the variability of the offset increasing rate, playback reboot events can be inferred.

The playback quality on a peer is largely determined by whether the peer can retrieve video chunks before their playback deadlines. Using buffer maps, for a specific chunk, we can calculate the fraction of peers that have retrieved the chunk at any time instant. The chunk retrieval ratio among peers should grow over time until it approaches 1. The speed of the growth is a good measure of the P2P video distribution efficiency. The chunk retrieval ratio is a good metric for service providers to make the capacity offload decisions. When this ratio is high, service providers may remove their capacity for other purposes; otherwise, service providers should add additional server capacity to sustain a high-level chunk retrieval ratio.

C. Optimization for Quality-of-Experience

The conventional Quality-of-Service (QoS) metrics, such as delay, loss, etc., which are commonly used in the network-layer service provisioning, are important but not sufficient in evaluating a successful IPTV application. We find that most current mesh-pull IPTV systems incur start-up delays

ranging from a few seconds to a couple of minutes, and the playback time lags among peers up to several minutes. Although research efforts have been conducted to minimize delays and increase the video bit rate for high-quality video services, IPTV service providers may optimize the IPTV application to achieve satisfied user experiences, namely, Quality-of-Experience (QoE).

In addressing noticeable start-up delay, some advertisement videos can be pre-cached on the users' side; when a user starts the IPTV application, immediately these advertisement videos are played back. During the playback of these commercials, videos are downloaded at the background. Therefore, users experience zero start-up delay, while the IPTV service provider finds an avenue to obtain revenues. Another possibility is to provide multiple videos with different bit rates for the same channel. Dedicated servers are deployed to provide the low-quality video and the P2P delivery architecture is utilized to transfer the high-quality video. When a user joins the service, the P2P engine downloads the low-quality video from the server to achieve a quick service initiation; then, the engine downloads the high-quality video from other peers. Technologies may provide the foundation of the IPTV application; nevertheless, a deep psychological understanding on user behaviors and expectations are helpful in optimizing the application to maximize the user viewing experience.

V. Traffic Pressure: TCP vs. UDP

The possible tidal wave of IPTV traffic will raise high pressure on the Internet. In a measurement study [2], the number of simultaneous users watching a live broadcast of the annual Spring Festival Gala on the Chinese New Year on January 28, 2006 reached over 200,000 users at bit rate in the 400-800 kbps range, corresponding to an aggregate bit rate in the vicinity of 100 gigabits/sec! In the future, we envision increasing IPTV traffic volume that thousands of streaming channels available on the Internet, each with a bit rate of 500 kbps or more, each supporting tens of users to hundreds of thousands of users.

The huge traffic volume generated by IPTV applications seems to be a growing concern among Internet Service Providers (ISPs) that need to support the distribution cost without sufficient incentives. Video chunks are the basic data units in mesh-pull systems. ISPs may consider to deploy caching infrastructures to cache these video chunks for assisting the peers within their domains and reducing the inter-ISP IPTV traffic. On the other hand, peers in mesh-pull streaming systems should also turn to more "locality-aware" [8] chunk delivery when selecting peers and video chunks for download.

The transmission of video chunks in mesh-pull streaming systems are carried using TCP or UDP. TCP provides a reliable and congestion aware transport. However, the selection of the TCP transport induces various implications on end-hosts, access links and network cores. Operating systems in general have constraints on the maximum rate to accept new TCP connections. When a peer has received a large number of TCP connection requests, there may not be a response for other TCP requests. Other networking applications running on the same host may suffer significantly, such as web browsing. Residential users may have additional difficulties since access routers at home is often of limited capacity and not capable to handle a large number of TCP connections. When one user is watching IPTV, other users using different computers within the same LAN may have a poor network connectivity. Recently we find that mesh-pull streaming systems tend to carry IPTV traffic using UDP instead of TCP. UDP incurs much less connection overhead than TCP; however, this increasing usage of UDP raises the concerns that these UDP traffic should be congestion-aware to avoid network collapse. Because UDP datagrams may be dropped in the networks, IPTV applications have to address how to react on packet loss. The simplest strategy is to ignore the dropped UDP datagrams with the expectation that the viewing quality does not suffer too much in spite of a small number of packet dropping; however, too much UDP segment loss will severely degrade the video playback performance. It may be beneficial to retransmit some lost UDP datagrams with the additional complexity in P2P applications. As a more promising alternative, these streaming traffic can be delivered over the Datagram Congestion Control Protocol (DCCP) [9] to achieve a good tradeoff between timeliness and reliability.

VI. Security Concerns: attacks and defenses

The distributed P2P architecture of mesh-pull streaming systems makes them prone to various security threats. A malicious peer in the system may mix video stream with bogus chunks, which may significantly degrade the quality of the rendered media at the receivers. This peer may also advertise a large number of non-existing peers who are interested in the same channel; therefore, a legitimate peer may find it difficult to identify other legitimate peers to download video chunks. Due to the low distribution cost of P2P streaming, we expect user-generated live video content (emanating from web cams and wireless devices) to be distributed using P2P mesh-pull architectures. The origin video server from users are mostly normal computers with limited CPU power and network capacity. If some malicious peers connect to this server and occupy its bandwidth without sharing the video chunks with other peers, other peers are not able to enjoy the video at all. A mesh-pull streaming system potentially consists of hundreds of thousands of peers. If malicious peers advertise that one victim host has abundant video chunks, other peers may send chunk requests to this victim host, consuming the CPU power and network bandwidth of this host. As a result, this victim host may undergo Denial-of-Service (DoS) attacks.

Similar attacks have been studied heavily in P2P file-sharing applications; nevertheless, few attacks have been reported for mesh-pull IPTV systems. Due to the real-time communication in IPTV, the potential attacks on mesh-pull system can be devastating. In [10], a chunk pollution attack is demonstrated to severely degrade the performance of an IPTV application. In the experiment, before launching the attack, a particular channel had about 3300+ viewers before the attack; during the attack the number of viewers dropped to about 500 within 30 minutes. The video quality became unacceptable for a

large majority of peers and they left the system eventually. In defending this attack, chunk signing is an effective mechanism. In chunk signing techniques, the so-called "authentication information", or signature, needs to be transmitted to the receivers along with the chunks. This authentication information can either be provided by the source (in which case the load on the source might be high) or could be distributed through the P2P system itself, in the form of a separate stream or be piggybacked with video chunks. A peer receives each chunk and its corresponding signature one by one, verifies its integrity and plays back (and forwards) only if the chunk is valid, otherwise rejects the chunk as being polluted. In facing the security challenges in mesh-pull streaming systems, researchers and developers should meet the real-time needs in defense. They may benefit from the lessons in combating the pollution, poisoning and DoS attacks in P2P file sharing.

VII. Architecture rethink: Tree-push, Mesh-pull, or Push-pull?

Mesh-pull systems have enjoyed a number of successful commercial deployments; however, tree-push systems have been largely running at the research stage. Nevertheless, those mesh-pull systems often suffer from long start-up delays, significant video switching delays and large peer playback time lags. Unlike mesh-pull systems, tree-push systems have smaller delay performance when the tree structure does not break down due to peer churn and peers at the higher level of the tree have sufficient upload capacity to support the streaming of their children peers. The debate on whether mesh-pull or tree-push is more suitable for IPTV is still on-going [11] [12]. The architecture design tradeoffs will impact various system characteristics significantly [13], i.e., system resilience and signaling overhead, etc..

In mesh-pull systems, the fundamental task of a peer is to download video chunks quickly. To this end, a peer should first locate its peer neighbors as potential chunk providers; then, identify which chunks to download. Either its neighbors notify this peer their buffer maps, or this peer inquires the buffer maps of its neighbors, so that the chunk availability information propagates among peers quickly. Finally, this peer "pulls" chunks from its neighbors according to the buffer map information. The amount of the time used in this process often contribute a large portion of the long service delay suffered in mesh-pull systems. Video can also be divided into chunks and be delivered via tree-push systems. Before the chunk delivery starts, the single/multiple-tree structure is established and the chunks are just forwarded from the root of the tree at the video source, to the different level of tree hierarchy. There are no further communication overhead for the subsequent chunk delivery unless the trees have to be repaired due to the departure of intermediate peers. Therefore, the delay performance in tree-push systems is in general good when the trees are stable.

Recently, a new type of push-pull architectures appears to be promising to offer a good tradeoff between the peer/chunk selection and scheduling overhead and the delay performance [12], [14]. The basic idea is to construct the chunk distribution trees between those peers that are stable and of high upload capacity, or super peers, so that video chunks can be pushed from the video source to these super peers quickly. Those ordinary peers, however, pull chunks from the super peers and other ordinary peers.

In this push-pull scenario, super peers contribute more upload capacity than ordinary peers; however, all the peers enjoy the same video playback quality. It is questionable on the incentives of one peer to become a super peer. One example to encourage peers' contribution is to provide differentiated service in IPTV applications. In [15], layered video can be used to provide such incentives among streaming peers. Ordinary peers only download a basic-layer video to receive the basic video playback quality; however, super peers download multiple-layer video to enjoy an enhanced video quality.

VIII. Conclusion

The current practice of mesh-pull P2P streaming systems demonstrate the feasibility of large-scale application layer multicast on top of the best-effort Internet. In this article, we provided an overview of existing mesh-pull systems. We briefly discussed several important design issues on performance, reliability and security of mesh-pull systems. Despite of the early successes, P2P IPTV is still at its early stage. Lots of open and interesting research problems remain to be addressed to design and deploy the next generation P2P IPTV services with superior user quality of experience. P2P IPTV systems should be enhanced with appropriate server infrastructure support to achieve a high level of quality-of-service. On the other hand, the popularity of P2P IPTV applications on the Internet has posed a great challenge for network infrastructures to support tremendous P2P IPTV traffic. It is therefore critical to design new P2P IPTV systems in a ISP-friendly fashion to minimize the traffic stresses which they imposed on ISP networks. In addition, the increasing user base of P2P IPTV systems will surely attract more malicious attacks. It is emergent to address various vulnerabilities in current designs to improve the security of future P2P IPTV systems.

Acknowledgment

This work was supported by the National Science Council under Grant ITR-0325726 and Grant CNS-0519998.

References

[1] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "DONet/CoolStreaming: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," in *IEEE INFOCOM*, vol. 3, Mar. 2005, pp. 2102 – 2111.

[2] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. on Multimedia*, vol. 9, no. 8, Dec. 2007.

[3] Y. Tang, J.-G. Luo, Q. Zhang, M. Zhang, and S.-Q. Yang, "Deploying P2P networks for large-scale live video-streaming service," *IEEE Comm. Mag.*, June 2007.

[4] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, "Will IPTV ride the peer-to-peer stream?" *IEEE Comm. Mag.*, June 2007.

[5] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer Internet video broadcast," *Proceedings of the IEEE*, 2007.

[6] X. Hei, Y. Liu, and K. W. Ross, "Inferring network-wide quality in P2P live streaming systems," *IEEE JSAC*, vol. 25, no. 10, Dec. 2007.

[7] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic fluid theory for P2P streaming systems," in *IEEE INFOCOMM*, 2007.
[8] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet service providers fear peer-assisted content distribution?" in *ACM IMC*, 2005.
[9] E. Kohler, M. Handley, and S. Floyd, "RFC4340: Datagram Congestion Control Protocol (DCCP)," March 2006.
[10] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena, "The pollution attack in P2P live video streaming systems: Measurement results and defenses," in *Sigcomm P2P-TV Workshop*, 2007.
[11] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live P2P streaming approaches," in *IEEE INFOCOMM*, 2007.
[12] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?" *IEEE JSAC*, 2007.
[13] V. Fodor and G. Dan, "Resilience in live peer-to-peer streaming," *IEEE Comm. Mag.*, June 2007.
[14] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast," in *IEEE ICDCS*, 2007.
[15] Z. Liu, Y. Shen, S. Panwar, K. Ross, and Y. Wang, "Using layered video to provide incentives in P2P streaming," in *Sigcomm P2P-TV Workshop*, 2007.