A Measurement Study of a Large-Scale P2P IPTV System

Xiaojun Hei[†], Chao Liang[‡], Jian Liang[†], Yong Liu[‡] and Keith W. Ross[†*]
[†]Department of Computer and Information Science
[‡]Department of Electrical and Computer Engineering
Polytechnic University, Brooklyn, NY, USA 11201
heixj@poly.edu, cliang@photon.poly.edu, jliang@cis.poly.edu, yongliu@poly.edu and ross@poly.edu

Abstract—**An emerging Internet application, IPTV, has the potential to flood Internet access and backbone ISPs with massive amounts of new traffic. Although many architectures are possible for IPTV video distribution, several mesh-pull P2P architectures have been successfully deployed on the Internet. In order to gain insights into mesh-pull P2P IPTV systems and the traffic loads they place on ISPs, we have undertaken an in-depth measurement study of one of the most popular IPTV systems, namely, PPLive. We have developed a dedicated PPLive crawler, which enables us to study the global characteristics of the mesh-pull PPLive system. We have also collected extensive packet traces for various different measurement scenarios, including both campus access network and residential access networks. The measurement results obtained through these platforms bring important insights into P2P IPTV systems. Specifically, our results show that 1.) P2P IPTV users have the similar viewing behaviors as regular TV users; 2) During its session, a peer exchanges video data dynamically with a large number of peers; 3) A small set of super peers act as video proxy and contribute significantly to video data uploading; 4) Users in the measured P2P IPTV system still suffer from long start-up delays and playback lags, ranging from several seconds to a couple of minutes. Insights obtained in this study will be valuable for the development and deployment of future P2P IPTV systems.**

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

EDICS category: Multimedia Streaming (5-STRM)

I. Introduction

With the widespread adoption of broadband residential access, IPTV may be the next disruptive IP communication technology. With potentially hundreds of millions of users watching streams of 500 kbps or more, IPTV would not only revolutionize the entertainment and media industries, but could also overwhelm the Internet backbone and access networks with traffic. Given this possible tidal wave of new Internet traffic, it is important for the Internet research community to acquire an in-depth understanding of the delivery of IPTV, particularly for the delivery architectures that hold the greatest promise for broad deployment in the near future.

There are several classes of delivery architectures for IPTV, including native IP multicast [1], application-level infrastructure overlays such as those provided by CDN companies [2], peer-to-peer multicast trees such as in end-system multicast [3], and mesh-pull P2P streaming such as CoolStreaming [4] and PPLive [5]. Each of these architectures classes imposes different traffic patterns and design challenges on Internet backbone and access networks. Requiring minimal infrastructure, P2P architectures offer the possibility of rapid deployment at low cost. An important characteristic of mesh-pull P2P systems is the lack of an (application-level) multicast tree - a characteristic particularly desirable for the highly dynamic, high-churn P2P environment [4].

In terms of the number of simultaneous users, the most successful IPTV deployments to date have employed mesh-pull P2P streaming architectures. Bearing strong similarities to BitTorrent [6], mesh-pull streaming deviates significantly from BitTorrent in various aspects:

1) BitTorrent in itself is not a feasible video delivery 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.
2) Due to the stringent need of video chunk availability before the deadline, fair resource sharing has not been carefully addressed in the current mesh-pull systems, in that, there have been no reciprocity mechanisms deployed in the current mesh-pull systems to encourage sharing between peers.
3) 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 thousands of users. Hence, gossip peer search algorithms have been equipped 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.

Several mesh-pull P2P streaming systems have been successfully deployed to date, accommodating tens of thousands of simultaneous users. Almost all of the these deployments have originated from China (including Hong Kong). The pioneer in the field, CoolStreaming, reported that more than $4,000$ simultaneous users in 2003. More recently, a number of second-generation mesh-pull P2P systems have reported phenomenal success on their Web sites, advertising tens of thousands of simultaneous users who watch channels at rates between 300 kbps to 1 Mbps. These systems include PPLive [5], PPStream [7], UUSee [8], SopCast [9], TVAnts [10], VVSky [11] and many more.

Given the success to date of many of these IPTV systems, as well as their potential to swamp the Internet with massive amounts of new traffic in the near future, we have been motivated to carry out an extensive measurement study on one of the mesh-pull P2P streaming systems, namely, PPLive. We chose PPLive as it is currently one of the most popular – if not the most popular – IPTV deployment to date. In particular, as part of a preliminary study we performed on PPLive, we measured the number of simultaneous users watching a PPLive broadcast of the annual Spring Festival Gala on Chinese New Year on January 28, 2006. We observed that PPLive broadcasted this event to 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 an earlier workshop paper, we reported preliminary measurement results for PPLive [12]. The current paper goes significantly further, providing a comprehensive study of PPLive, including insights into the global properties of the system. Achieving these deeper insights has been challenging because the PPLive protocol is proprietary. In particular, in order to build the measurement tools that were used to collect much of the data in this paper, we had to analyze a large portion of the PPLive protocol.

In this paper, we seek to answer the following questions about a large-scale P2P IPTV deployment:

- *What are the user characteristics?* For both popular and less-popular PPLive channels, how does the number of users watching a channel vary with time? As with traditional television, are there diurnal variations in user demand? What are the dynamics of user churn? What is the geographic distribution of the users, and how does this distribution fluctuate over time.
- *How much overhead and redundant traffic is there?* What fraction of bytes a peer sends (or receives) is control data and what fraction is actual video data? What fraction of the video traffic that a peer receives is redundant traffic?
- *What are the characteristics of a peer's partnerships with other peers?* How many partners does a peer have? What are the durations of the partnerships? At what rates does a peer download from and upload to its partners? How are the partnerships different for a campus peer and a residential peer? How do the partnerships compare to those in BitTorrent?
- *What are the fundamental requirements for a successful mesh-pull P2P IPTV system?* How does a P2P IPTV system maintain high enough downloading rates on all peers with heterogeneous uploading capacities? What is the video buffering requirement for smooth playback on individual peers in the face of rate fluctuations on peering connections and peer churns?

We attempt to answer these questions by using a custom-designed PPLive crawler and using packet sniffers deployed at both high-speed campus access and broadband residential access points. Quantitative results obtained in our study bring light to important performance and design issues of live streaming over the public Internet.

Using our previous work in [12] and [13] as a base, we present a comprehensive active and passive measurement study of PPLive. Our contributions are as follows:

- For active crawling, we refine our peer tracking methodology, originally proposed in [13], to accurately trace the dynamics of peers behind NAT/firewalls.
- Our crawling results include both channel-level peer measurement as well as all-channel peer statistics at different time-scales, i.e., day-level and week-level.
- Our crawling apparatus also includes a buffer map crawling component. Buffer maps reflect the video content cached by peers and the playback process implicitly. These measurement results are reported in this paper to demonstrate the significant playback time lag between PPLive peers. This finding brings forth the necessity to carefully design video chunk scheduling schemes to minimize this playback lag for live streaming.
- We present passive sniffing results with numerous improvements (i.e., refine the video traffic filtering rules so that we are able to determine video traffic exchange more accurately). Our measurement results show the traffic pattern and peer dynamics of PPLive users. These findings provide insights for ISPs to conduct appropriate traffic engineering, for example, track P2P streaming traffic, design new streaming caching schemes, and so on.

This paper is organized as follows. In Section II, we provide an overview of different aspects of mesh-pull streaming systems including architecture, signal and management protocols based on our measurement studies. Our measurement tools include an active crawler and a passive sniffer. In Section III, using our PPLive crawler, we present the global-scale measurement results for the PPLive network, including number of users, arrival and departure patterns, and peer geographic distributions. We also provide a qualitative characterization of the delay performance of the PPLive streaming service in Section IV. In Section V, by sniffing monitored peers, we present the traffic patterns and peering strategies as viewed by residential and campus PPLive clients. We provide an overview of the related P2P measurement work in Section VI. Finally, based on our measurement results, we outline some design guidelines for the successful deployment of IPTV application over the Internet in Section VII.

II. Overview of mesh-pull P2P streaming systems

Current mesh-pull streaming systems provide little information about their proprietary technologies. Through our measurement studies and protocol analysis on two well-known mesh-pull streaming systems, PPLive and PPStream, we have gained significant insights into the protocols and streaming mechanisms of mesh-pull streaming systems. In order to gain a better understanding of our measurement tools and results, in this section we provide an overview of a generic mesh-pull system. Figure 1 depicts a generic mesh-pull P2P live streaming architecture. There are three major components in the mesh-pull streaming architecture:

1) The streaming peer node includes a streaming engine and the media player, co-located in the same machine. All the peers cooperatively deliver video chunks among themselves from the channel streaming server via the streaming engine. The streaming engine download media chunks from other peer nodes or the channel streaming server; these chunks are reassembled into the original media content and stream to the media player for playback.
2) The channel stream server converts the media content into small video chunks for efficient distribution among peers.
3) The tracker server provides streaming channel, peer and chunk information for each peer node to join the network and download video chunks from multiple peers in the system requesting the same media content.

A mesh-pull P2P live streaming software, running in user computers (peers), typically has two major communication protocols: (i) a peer registration, channel and peer discovery protocol; and (ii) a P2P media chunk distribution protocol. Figure 2 depicts an overview of the peer registration, channel and peer discovery protocol. When an end-user starts the chunk-pull streaming software, it joins the network and becomes a streaming node. The first action (step 1) is to download a list of channels distributed by the streaming network from the tracker server. Once the user selects a channel, this peer node register itself in the tracker server and requests an initial list of peers that are currently watching the same channel. The peer node then communicates with the peers in the list to obtain additional lists (step 2), which it aggregates with its existing peer list. In this manner, each peer maintains a list of other peers watching the channel. A peer on a list is identified by its IP address and UDP and TCP signaling port numbers. The registration and peer discovery protocol is commonly running over UDP; however, if UDP fails (for example, because of a firewall), TCP may also be used for registration and peer discovery. Utilizing this distributed gossip-like peer discovery protocol, the signaling overhead at the tracker server is considerately reduced; hence, a small number of tracker servers are able to manage possibly millions of streaming users.

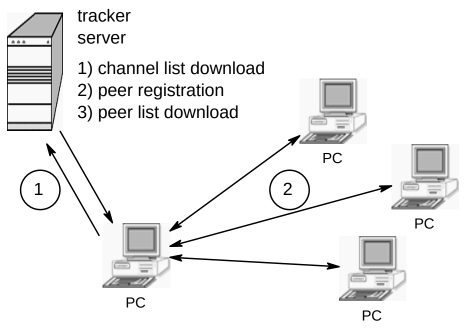


Fig. 2. Channel and peer discovery

We now describe the chunk distribution protocol. At any given instant, a peer buffers 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 upload chunks to 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. The buffer map message includes the offset (the ID of the first chunk), the length of the buffer map, and a string of zeroes and ones indicating which chunks are available (starting with the chunk designated by the offset). If the offset field is of 4 bytes, for one channel with the bit rate of 340 kbps and a chunk size of 14 Kbytes, this chunk range of 2^{32} indicates the time range of 2042 days without wrap-up. The **BM width** is the difference between the newest and oldest chunk number advertised in a buffer map message. The **BM playable video** is the number of contiguous chunks in the buffer map, beginning from the offset. Figure 3 illustrates a buffer map.

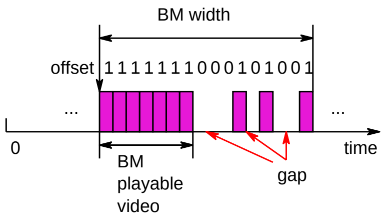


Fig. 3. A peer's buffer map of video chunks

A peer can request, over a TCP connection, a buffer map from any peer in its current peer list. After peer A receives a buffer map from peer B, A can request one or more chunks that peer B has advertised in the buffer map. 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. Clearly, when a peer requests chunks, it should give some priority to the missing chunks that are to be played out first. Most likely, it also gives priority to rare chunks, that is, chunks that do not appear in many of its partners' buffer maps (see [14] [4] [13]). Peers can also download chunks from the original channel server. The chunks are usually sent over TCP connections. In some mesh-pull systems, video chunks are also transferred using UDP to achieve timely delivery.

Having addressed how chunks are distributed among peers, we now briefly describe the video display mechanism. As mentioned above, the streaming engine works in conjunction with a media player (either Windows Media Player or RealPlayer). Figure 4 illustrates the interaction between the streaming engine and the media player. The streaming engine, once having buffered a certain amount of contiguous chunks, launches the media player. The media player then makes an HTTP request to the engine, and the engine responds by sending video to the media player. The media player buffers the received video; when it has buffered a sufficient amount of video content, it begins to render the video.

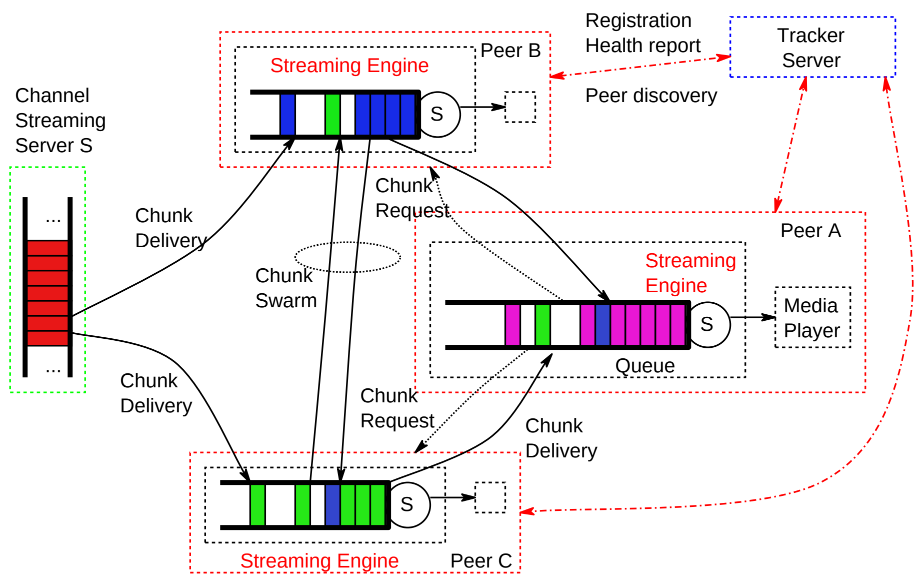


Fig. 1. Mesh-pull P2P live streaming architecture

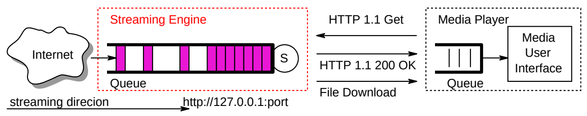


Fig. 4. Streaming process of mesh-pull systems

If, during video playback, the streaming engine becomes incapable of supplying the video player with data at a sufficient rate (because the client is in turn not getting chunks fast enough from the rest of the network), then the media player will starve. When this occurs, depending on the severity of the starvation, the streaming engine may have the media player wait where it left off (freezing) or it may have the media player skip frames.

PPLive is a typical mesh-pull P2P streaming system. We now provides a brief introduction on PPLive. PPLive is a free P2P IPTV application. According to the PPLive web site [5] in May 2006, PPLive provides 200+ channels with $400,000$ daily users on average. The bit rates of video programs mainly range from 250 kbps to 400 kbps with a few channels as high as 800 kbps. PPLive does not own video content; the video content is mostly feeds from TV channels, TV series and movies in Mandarin. The channels are encoded in two video formats: Window Media Video (WMV) or Real Video (RMVB). The encoded video content is divided into chunks and distributed to users through the PPLive P2P network.

Our P2P network measurements on PPLive fall into two categories: active crawling and passive sniffing. The active crawling is used to obtain user behaviors and global view of the entire PPLive network for any channel. The passive sniffing is used to gain a deeper insight into PPLive from the perspective of residential users and campus users.

III. Global View of User Behavior

We are interested in IPTV user behavior characteristics, such as the evolution of the number of active users within a channel, the evolution of the number of active users aggregated across all channels, user arrival and departure patterns, the distribution of channel popularity, and the geographic location of users. To understand user behavior across the entire PPLive network, we developed a crawler which continuously tracks all participating peers, where a peer is identified by a combination of IP address and TCP/UDP service port number.

A. *Peer Tracking Methodology*

Peer tracking is a challenging problem since we do not have direct access to the proprietary PPLive tracking servers. Our methodology for peer tracking exploits PPLive's distributed gossiping algorithm. Recall that each individual peer maintains a list of active peers that it knows about. Further, each peer can update its list by retrieving lists from its neighboring peers. Our methodology does the following for each channel:

- Time is divided in rounds of T seconds. At the beginning of each round, we request peer lists from multiple peer-list servers. The retrieved lists are merged into an initial aggregate list. Initially, all peers on the list are marked as "uninvestigated".
- For each uninvestigated peer on this aggregate list, we request its peer list and then mark the peer as "investigated". The retrieved list are merged into the aggregate peer list. We continue this crawling process, retrieving and merging lists from uninvestigated peers. This crawling process continues for the first S seconds (with $S \leq T$) of each round. At the end of the S seconds, we save the aggregate list, clear the aggregate list, sleep for $T - S$ seconds, and then begin another round.
- We thus generate a sequence of aggregate lists, one for each T-second crawling period. Denote the ith aggregate list as L_i, $i = 0, 1, 2, \ldots$. For a given peer p, we define its "list-joining time" as time $S + i_p T$, where L_{i_p} is the first aggregate list on which the peer is recorded. Similarly, we define its "list-leaving time" as time $S + j_p T$, where

L_{j_p} is the first aggregate list after i_p for which p is no longer present.

- We draw conclusions about user behavior from the sequence of aggregate peer lists, as well as from the list-joining and list-leaving times.

The methodology described above, although promising, has a number of issues that need to be addressed. First, how do we obtain a peer list from a peer or from a peer-list server? Second, when a peer truly joins the channel, will it eventually appear on an aggregate list? If so, how much of a lag is there from when it joins the channel until the list-joining time? Third, when a peer quits the channel, how much time elapses from when it quits until the list-leaving time?

To generate the sequence of aggregate peer lists for a channel, we developed a PPLive crawler. This task in itself was challenging we needed to implement portions of the PPLive proprietary protocol. To this end, using packet traces from passive sniffing and our knowledge about how mesh-pull P2P streaming systems generally operate, we were able to understand critical portions of PPLive's signaling protocols. With this knowledge, we were then able to send PPLive peer-list request messages to the PPLive peers. Specifically, as shown in Figure 5, the crawler operates as follows:

- *Peer Registration:* The crawler first registers itself with one of the root servers by sending out a peer registration message. The significant information in this message includes a 128 bit channel identifier, its IP address, and its TCP and UDP service ports. In contrast to many other popular P2P applications, a PPLive peer does not maintain a fixed peer ID, but instead creates a new, random value every time it re-joins the channel.
- *Bootstrap:* After the registration, the crawler sends out one bootstrap peer list query message to each peer-list root server for retrieving an initial peer list for this channel. In response to a single query, the server returns a list of peers (normally 50 peers), including IP addresses and service port numbers. The crawler aggregates all the lists it has received from the server, thereby maintaining an initial list of peers enrolled in the channel.
- *Peer Query:* The crawler then sends out queries to the peers already on the list to ask for their peer lists. Peer lists returned by those peers are merged into the crawler's current peer list. (Peers behind NATs often do not respond to peer-list queries, as we will discuss below.)

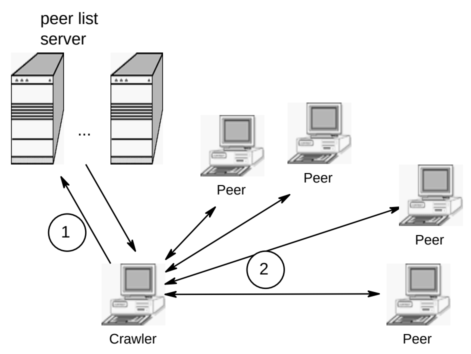


Fig. 5. PPLive peer list crawling

The number of discovered peers by the crawler for one channel over 25 seconds is plotted in Figure 6. It shows that the crawler can find 95% of peers for a channel within 5 seconds. Given this observation, we set the round length to $T = 60$ seconds and crawling duration within a round to $S = 15$ seconds.

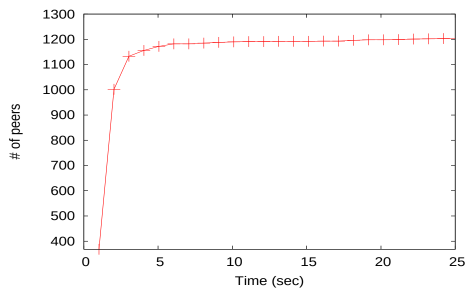


Fig. 6. The number of peers on the aggregate peer list collected by the crawler for one channel

We now investigate how accurately an aggregate list reflects the actual peers participating in a channel. One possibly way to verify whether a peer on the aggregate list is actually an active peer is to probe the peer by querying it for its peer list. Unfortunately, a large fraction of PPLive peers (more than 50%) are behind NATs and do not respond to contact initiated from the outside. Since probing is not viable, we instead performed the following experiment to investigate the accuracy of the aggregate peer list:

1) For a given channel, we start the crawler with rounds of $T = 60$ seconds.
2) We have a peer under our control join the channel. We record the time s when it connects to the channel.
3) We determine the time t when this controlled peer first appears on one of the aggregate lists L_i. We define the *arrival lag* as $t - s$.
4) Subsequently, we remove the controlled peer from the channel and record the time u.
5) We determine the time v when the controlled peer no longer appears in the aggregate lists. We define the *departure lag* as $v - u$.

We repeated this experiment under a variety of conditions, with the controlled peer being behind a NAT in half of the experiments. In each experiment, the controlled peer was eventually observed on an aggregate list. Therefore, we can safely assume that every peer that joins a channel eventually appears on an aggregate list. Also in each experiment, once the controlled peer was removed from the channel, it eventually disappeared from the aggregate lists. Therefore, we can safely assume that after a peer leaves a channel, it eventually departs from the aggregate list. On average over 33 experiments, a peer's arrival lag was 31.6 seconds; a peer's departure lag was 104.2 seconds. One explanation for the faster detection of peer arrivals is that a peer leaves the aggregate list only after it disappears from the peer lists of all peers, whereas a peer arrival can be detected as long as it appears on a peer list returned by the peer-list root server or any other peer. There are several implications of the results of this experiment:

- The arrival and departure curves generated from the sequence of aggregate lists reflect the actual arrival and departure rate curves, but include random lags. The lags are small compared to the time-scale we are interested in.
- Due to the mismatch between the arrival and departure lags, a peer's sojourn time inferred from the aggregate lists is, in average, around 70 seconds longer than its actual value. This has two consequences. First, it skews the peer lifetime distribution. However, since the measured average sojourn times for different channels are in the range of $800 \sim 1600$ seconds, the skew is minor. Second, it overestimates the number of active peers. From Little's law, the overestimate ratio can be calculated as $\frac{70}{X}$, where X is the real average peer sojourn time measured in seconds. Consequently, the active peer numbers we subsequently report overestimate the real active peer numbers by $5 \sim 9\%$.

B. Evolution of Participating Users

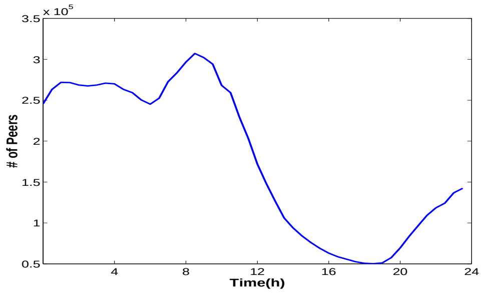
We would like to understand how the number of P2P IPTV users evolve over time. We explore peer participation evolution for all channels, for popular and unpopular TV channels, and for a popular movie channel. The time in the figures are labeled in US Eastern Standard Time (GMT-5). Figure 7 shows the total peer participation evolution for one day and one week, aggregated over all the PPLive 400+ channels. The diurnal trend is clearly demonstrated in Figure 7(a). The major peaks appear during 8AM to 12PM EST, translating into 8PM to 12AM China local time (GMT+8). As we shall see, those peaks are mostly contributed by users from China. (We remind the reader that our crawling methodology slightly overestimates the number of participating peers.)

In Figure7(b), for the week of Feb. 12 18th, 2007, the daily PPLive users remain with a constant diurnal viewing pattern. We also observed similar trends in other monitored weeks. This might suggest that IPTV users have different profiles and viewing habits than regular TV/movie audiences.

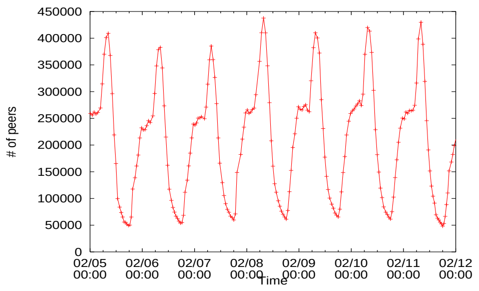
One important characteristic of IPTV is its ability to reach a global audience. A traditional broadcast television channel typically offers its most popular content when its local audience is most available (for example, in the evening). But because an IPTV service can reach a global audience in all time zones (see Figure 15 and Table IV), there will be incentives to offer popular content more evenly throughout the 24-hour day. Thus, in the future, we may see a flattening of the diurnal patterns.

We ranked each of the crawled channels according to their peak number of participating peers. In Figure 8 we plot the peak number of participating peers versus channel rank in the log-log scale. We observe a clear linear relationship (with a slope 0.41) between peer population vs. channels, especially for the top 100 channels.

Figure 9 shows how the number of participating users evolves for a popular and a less-popular TV channel. We first observe that the numbers of participating peers are quite different for the two programs. The maximum number of peers for the popular program reaches nearly $2,700$; however, that



(a) One day (Oct. 3, 2006)



(b) One week (Feb. 5 to Feb. 12, 2007)

Fig. 7. Evolution of total number of peers in the PPLive network

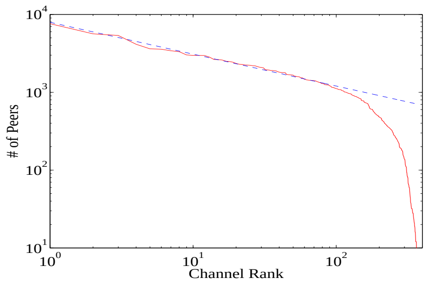
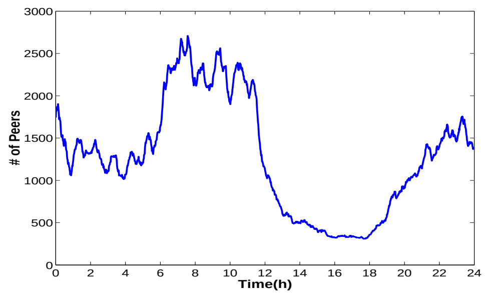


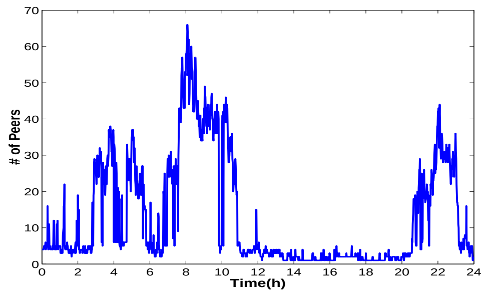
Fig. 8. Distribution of peak number of peers among all channels at EST 08:30AM Oct.03, 2006

of the unpopular program is just around 65. The major peaks appear during 7AM to 12PM EST, translating into 7PM to 12AM China local time. This suggests that people tend to use IPTV to watch TV programs outside of office hours, consistent with the behavior of regular TV users. In contrast, a recent measurement study on Skype [15] suggests that people tend to use VoIP service at work.

As with many other P2P applications, the number of IPTV users is largely determined by the popularity of the program. The annual Spring Festival Gala on Chinese New Year is one of the most popular TV programs within Chinese communities all over the world. Starting from 3AM EST, January 28, 2006 (Chinese New Year Eve day), we ran the crawler to collect all peer IP addresses from 14 PPLive channels which were broadcasting the event. Figure 10 plots the number of peers watching this event live through PPLive. There was a sharp jump from $50,000$ peers to $200,000$ peers after the event started at 7AM EST. The number of users held at this high level for around 4 hours. The number of users went back to normal when the event finished at about 11AM EST. The near constant user population during the event suggests that mesh-



(a) Popular TV channel



(b) Unpopular TV channel

Fig. 9. Diurnal trend of number of participating users on Oct. 13, 2006

pull P2P streaming systems scales well, handling a flash crowd in a live broadcasting.

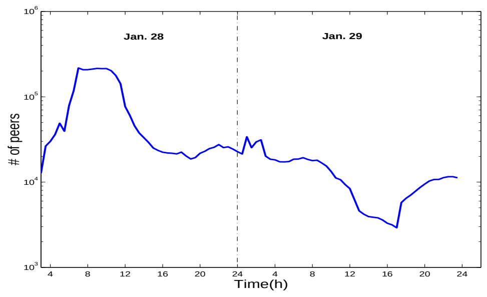
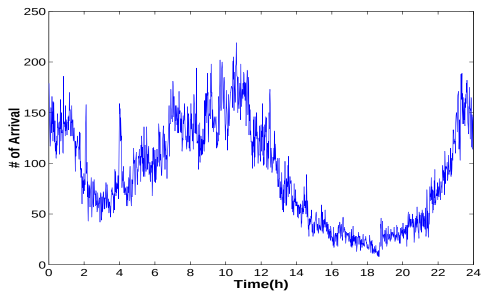


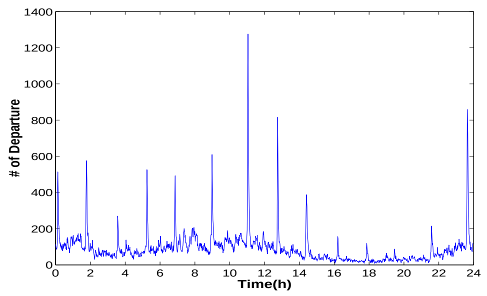
Fig. 10. Flash crowd on Chinese new year eve

C. User Arrivals and Departures

In this section, we examine the peer arrival and departure pattern for various PPLive channels. We plot the numbers of peer arrivals and departures of the popular movie channel in every minute of one day in Figure 11. Comparing it with the evolution of the number of participating peers, we find that peers join and leave at a higher rate at peak times. We also see consecutive spikes with a period of about 2 hours in the departure rate curve in Figure 11(b). The spikes are due to many peers leaving immediately and simultaneously at the end of (roughly) two-hour programs. This batch-departure pattern in P2P IPTV systems is different from P2P file sharing systems, where peer departures are mostly triggered by the asynchronous completions (or, the detections of completions) of file downloads. This suggests that P2P IPTV systems expect lower peer churn rates in the middle of a program. Consequently, peers can maintain more stable partnership with each other. We will address this peer dynamics more in Section V-D3.



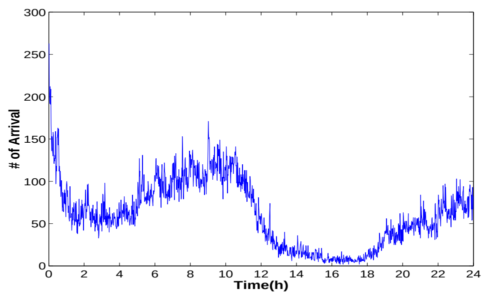
(a) Peer arrival rate



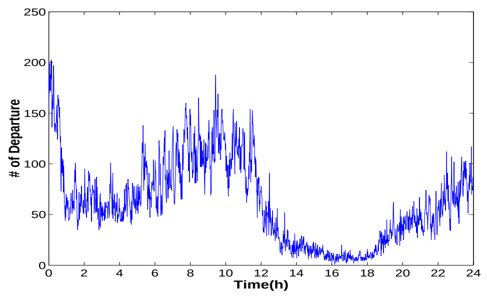
(b) Peer departure rate

Fig. 11. Peer arrival and departure evolution of a popular movie channel

We also plot the numbers of peer arrivals and departures of the popular TV channel in every minute of one day in Figure 12. We observe that the arrival pattern of this TV channel is similar to that of the movie channel. However, there is no periodic batch departure pattern for this TV channel.



(a) Peer arrival rate



(b) Peer departure rate

Fig. 12. Peer arrival and departure evolution of a popular TV channel

We define the peer lifetime as the time between the arrival and the departure of the peer. Our analysis shows that peer lifetimes vary from very small values up to 16 hours. There are totally $34,021$ recorded peer sessions for the popular channel and $2,518$ peer sessions for the unpopular channel. The peer lifetime distribution in Figure 13 suggests that peers prefer to stay longer for popular programs than for unpopular programs. However 90% of peers for both programs have lifetimes shorter than 1.5 hours.

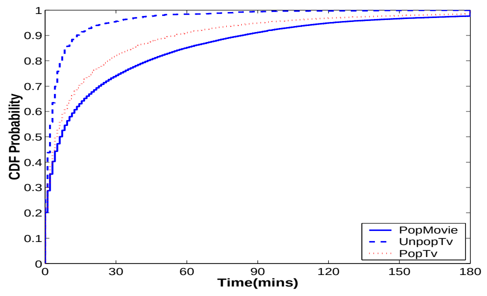


Fig. 13. Peer lifetime distribution

D. User Geographic Distribution

We classify PPLive users into three regions: users from Asia, users from North America, and users from the rest of the world. To accomplish this classification, we map a peer's IP address to a region by querying the free MaxMind GeoIP database [16]. Figure 14 shows the evolution of the geographic distribution of the popular channel during one full day. The figure is divided into three regions by two curves. The bottom region is made up of the peers from Asia, the middle region is for the peers from North America, and the top region is for peers from the rest of the world. We can see that most of users come from Asia. Again, the percentage of peers from Asia reaches the lowest point around 7PM to 8PM EST.

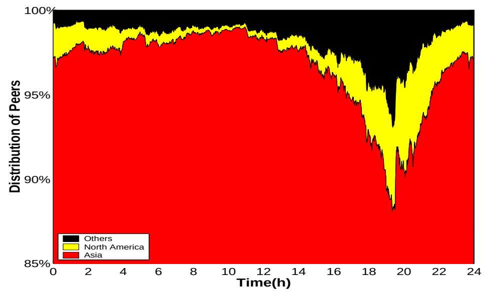


Fig. 14. Geographic distribution of popular movie channel

Fig 15 plots the evolution of peer geographic distribution for the Spring Festival Gala event on the past Chinese New Year's Eve. This figure has the same format as Figure 14, with three regions denoting three different geographical regions. We can see that for this event, many peers from outside of Asia were watching this live broadcast – in fact, a significantly higher percentage of peers were from outside of Asia as compared with Figure 14. The geographic distribution evolution is consistent with the observations in Section III-C: Peers from North America have the smallest share at about 7AM EST, and the largest share at about 8PM EST. Thus the behavior of users in North America is quite similar to users in Asia.

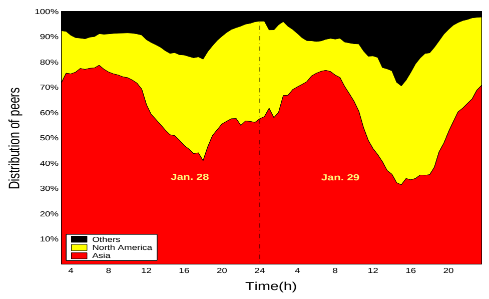


Fig. 15. Evolution of geographic distribution during Chinese new year's eve

One lesson learned from this crawling study is that, with P2P IPTV systems, it is possible to track detailed user behavior. Unlike traditional broadcast television, the operators of a P2P IPTV system can track the type of programs a user watches, the region in which the user lives, the times at which the user watches, and the users channel switching behavior. Such detailed information will likely be used in the future for targeted, user-specific advertising. This research also demonstrates that an independent third-party can also track peer and user characteristics for a P2P IPTV system. Similar to file-sharing monitoring companies (such as Big Champagne [17]), IPTV monitoring companies will likely emerge, and will provide content creators, content distributors, and advertisers with information about user interests.

IV. Playback Delay and Playback Lags among Peers

As theoretically demonstrated in [18], appropriate buffering can significantly improve video streaming quality. However, too much buffering may make the delay performance unacceptable for a streaming service. In this section, we report *quantitative* results on the buffering effect of PPLive peers on the delay performance. In particular, we used our measurement platforms, to obtain insights into start-up delay and playback lags of peers.

A. Start-up Delay

Start-up delay is the time interval from when one channel is selected until actual playback starts on the screen. For streaming applications in the best-effort Internet, start-up buffering has always been a useful mechanism to deal with the rate variations of streaming sessions. P2P streaming applications additionally have to deal with peer churn, increasing the need for startup buffering and delay [18]. While short start-up delay is desirable, certain amount of start-up delay is necessary for continuous playback. Using our monitored peers (see Section V), we recorded two types of start-up delays in PPLive: the delay from when one channel is selected until the streaming player pops up; and the delay from when the player pops up until the playback actually starts. For a popular channel, the measured player pop-up delay was from 5 to 10 seconds and the player buffering delay was 5 to 10 seconds. Therefore, the total start-up delay was from 10 to 20 seconds; however, less popular channels had start-up delays of up to 2 minutes. These delays are, of course, significantly longer than what are provided by traditional television. Hence, the current state-of-the art of mesh-pull P2P streaming technology does not provide users with the same channel-surfing experience as traditional television.

B. *Video Buffering*

As illustrated in Section II, in mesh-pull systems, peers exchanges video chunk information between themselves using buffer maps. To monitor the buffer content of PPLive clients, we augmented the crawler with a TCP component that retrieves the buffer map from the peers during the peer list crawling process, as shown in Section III-A. As we crawl each peer, the crawler sends a PPLive request for the peer's buffer map. We then parse the buffer maps off-line, to glean information about buffer resources and timing issues at remote peers throughout the PPLive network.

In this subsection, we quantify buffer levels across peers actively watching a specific channel. As illustrated in Figure 3, a buffer map returned by a peer not only indicates how many video chunks are buffered at that peer (the number of 1s in the buffer map), but also indicates how many video chunks can be played continuously on that peer when the buffer map was returned to the crawler (the number of consecutive bit-1s at the left side of the buffer map). We plot in Figure 16 the CDF of the buffer levels among 200+ peers over a 600-minute crawling period for a gaming channel. Both the total buffer levels and continuous playable buffer levels are plotted.

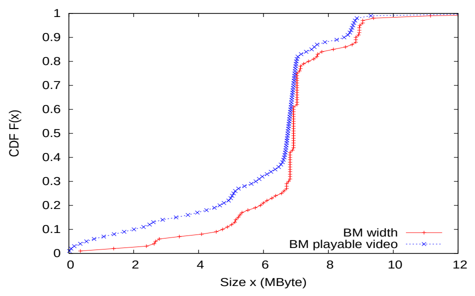


Fig. 16. CDF of buffer level for a game channel among 200+ peers over a 600-minute crawling period.

In Figure 16, we observe that peers seem to strive for buffer levels of 7 Mbytes or higher. With only a small probability peer buffers went under 1 Mbytes.

C. *Playback Lags among Peers*

One unfortunate characteristic of a mesh-pull P2P streaming system is the possibility of playback lags among peers due to the deployment of the buffering mechanisms. Specifically, some peers watch frames in a channel minutes behind other peers. Thus, for example, in a soccer game some peers will see a goal minutes after other peers. Additionally, peers with large playback lags won't upload useful chunks to peers with smaller lags, decreasing the aggregate uploading capacity of the system.

To analyze this lagging effect, we again use the buffer maps harvested from our crawler. Recall that each buffer map includes an offset, which provides the earliest chunk buffered in the PPLive engine. This offset increases along with the playback. We use the buffer map offset as a reference point of the actual playback. Therefore, the lags of buffer map offsets among peers watching the same channel reflect the lags of the actual playbacks among them. We intensively probed peers participating in a specific TV channel, ATV, to retrieve their buffer maps for 40 minutes. We clustered the harvested buffer maps according to the time when they are received by the crawler. Received buffer maps are clustered into time bins of 5 seconds. For buffer maps within each bin, we calculated the difference between the maximum and minimum offset. Figure 17 plots the maximum playback time differences over all bins. We observe that the lag among peers can be huge - with around 35 probed peers within one bin, the maximum playback time difference of these peers are as high as 140 seconds.

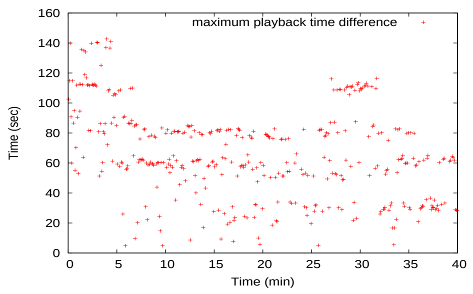


Fig. 17. Maximum playback time difference for the ATV channel over a 40-minute period.

V. Connection and Traffic Characteristics

In this section we explore connection and traffic characteristics of mesh-pull P2P streaming systems. We explore degree of packet redundancy, download and upload traffic levels, properties of TCP connections, and whether the underlying traffic flows are similar to those of BitTorrent or tree-pull P2P streaming systems.

To this end, we sniffed and collected multiple PPLive packet traces from four PCs: two PCs connected to Polytechnic University campus network with 100 Mbps Ethernet access; and two PCs connected to residential networks through cable modem. Most of the PPLive users today have either one of these two types of network connections. The PCs with residential access were located at Manhattan and Brooklyn. Each PC ran Ethereal [19] to capture all inbound and outbound PPLive traffic. We built our own customized PPLive packet analyzer to analyze the various fields in the various PPLive signaling and content packets.

As summarized in Table I, we collected traces from four peers, each of which was watching one of two channels (either the popular CCTV3 or the less popular CCTV10) from either a campus network or residential networks. Data were obtained at different granularities, including byte-level, packet-level and session-level, to help us understand PPLive's signaling and streaming protocols and its impact on the Internet.

TABLE I
DATA SETS

Trace Name	Trace size (Byte)	Duration (Sec)	Playback Rate (Kbps)	Download (Mbyte)	Upload (Mbyte)
CCTV3-Campus	784,411,647	7676	340	360.99	4574.57
CCTV3-Residence	132,494,879	7257	340	372.53	352.75
CCTV10-Campus	652,000,813	7285	312	317.08	3815.34
CCTV10-Residence	66,496,909	9216	312	385.50	7.68

A. Methodology for Isolating Video Traffic

A PPLive peer generates and receives both video and signaling traffic. In this paper, we are mainly concerned with the video traffic, since it is responsible for the majority of the traffic in most P2P streaming systems. In order to present a clear picture of the nature of the PPLive video traffic, we use a simple heuristic to filter out the signaling traffic from our traces. The ideas behind heuristic can likely be employed for the analysis of many P2P streaming systems, including PPLive.

In a mesh-pull P2P video streaming system, a peer normally has a large number of ongoing TCP connections with other peers. Some of these connections contain only signaling traffic; other connections contain video chunks and possibly some signaling traffic. The chunk size is typically much larger than the maximum payload size of a TCP segment (typically 1460 bytes). For example, in PPLive, the chunk size is larger than 14 Kbytes (the exact chunk size depends on the bit rate). Thus, if a TCP connection carries video, it should have a large number (say, at least 10) of large size TCP segments (say, $>$ 1200 bytes) during its lifetime. These observations lead to the following heuristic:

1) For a given TCP connection, we count the cumulative number of large packets ($>$ 1200 bytes) during the connection's lifetime. If the cumulative number of large packets is larger than 10, this connection is labeled as a "video TCP connection"; otherwise, the connection is labeled as a "signaling TCP connection". We filter out all signaling TCP connections from the traces.
2) A video TCP connection may include some signaling traffic as well. For each video TCP connection, we further filter out all packets smaller than 1200 bytes.

We first use this heuristic to estimate the fraction of upstream and downstream signaling overhead for each of the four traced peers. The signaling overhead consists of the payloads of all UDP packets, plus the payloads of all the TCP packets in the signaling connections, plus the payloads of all the TCP packets less than 1200 bytes in all of the video connections. From Table II we see that the signaling overhead is generally in the 5% to 8% range except for the upload traffic in trace CCTV10-Residence. The reason is that the uploading video traffic in that trace is extremely low.

B. Video Traffic Redundancy

Due to the distributed nature of mesh-pull P2P streaming, it is possible that a peer downloads duplicate chunks from multiple partners. The transmission of redundant chunks wastes network and access bandwidth; hence, we are interested in measuring the traffic redundancy after the streaming player begins to playback steadily. To this end, to minimize the impact of transient behavior, the first 10 minutes of the traces are not used for this redundancy analysis . Excluding TCP/IP headers, we determine the total streaming payload for the download traffic. Utilizing the video traffic filtering heuristic rule, presented in Section V-A, we are able to extract the video traffic. Given the playback interval and the media playback speed, we obtain a rough estimate of the media segment size for the playback interval. We compute the redundant traffic as the difference between the total received video traffic and the estimated media segment size. We define the redundancy ratio as the ratio between the redundant traffic and the estimated media segment size. From Table III, we observe that the traffic redundancy is small. This is partially due to the long buffer time period so that PPLive peers have enough time to locate peers in the same streaming channel and exchange content availability information between themselves.

The negative redundancy ratio (-3.5%) for CCTV3-Campus indicates that the video download chunks are not sufficient for smooth video playback. As shown in Figure 18(a), at time $10 < t < 20$ minute and $60 < t < 64$ minute for CCTV3-Campus, the download rate decreases significantly and the PPLive playback may suffer seriously lacking of video chunks. Given the good connectivity of campus network, this abnormal case requires further investigation.

C. Download and Upload Video Traffic

Having isolated the video traffic, we examine the aggregate amount of upload and download video traffic leaving and entering the four peers. Figure 18 plots the upload and download rates of the video traffic for the four traces beginning from startup. Each data point is the average bit rate over a 30 second interval. We make the following observations:

1) The aggregate download rates closely hug the video playback rates, even for campus peers where the available bandwidth greatly exceeds the playback rate. This is very different from BitTorrent, which tries to use as much of its downstream bandwidth as possible.
2) A P2P streaming peer's aggregate upload rate can greatly exceed the aggregate download rate. For example, we see that for two campus peers, the upload rate exceeds the download rate by (approximately) a factor of 10. This is also very different from BitTorrent, whose tit-for-tat mechanism encourages peers of roughly equal capacity to partner with each other.
3) In a P2P streaming system, not all peers have an aggregate upload rate exceeding the download rate. For example, we see that one of the residential peers

TABLE II
PPLIVE TRAFFIC OVERHEAD

Trace name	upload (Mbyte)				download(Mbyte)			
	Signaling		Video	Percentage	Signaling		Video	Percentage
	UDP	TCP	TCP	Overhead(%)	UDP	TCP	TCP	Overhead(%)
CCTV3-Campus	0.60	43.7	3951.4	1.11	0.88	22.6	285.7	7.59
CCTV3-Residence	2.87	19.9	313.2	6.78	4.40	23.5	314.9	8.14
CCTV10-Campus	1.23	73.8	3406.3	2.16	1.28	21.6	259.4	8.11
CCTV10-Residence	1.55	2.7	4.4	49.13	2.76	23.9	351.6	7.05

TABLE III
VIDEO TRAFFIC REDUNDANCY

Trace name	Interval (second)	Total download (Mbyte)	Video download (Mbyte)	Estimated media segment size (Mbyte)	Redundancy ratio
CCTV3-Campus	6966.2	308.3	285.7	296.1	-3.5%
CCTV3-Residence	6512.6	338.4	314.9	276.8	13.8%
CCTV10-Campus	6600.7	281.0	259.4	257.4	0.76%
CCTV10-Residence	8230.5	375.5	351.6	321.0	9.5%

uploads at an average rate approximately equal to the average download rate, and the other residential peer does almost no uploading. Thus, some peers act as *amplifiers*, pumping out bytes at rates higher than the receive rate; some peers act as *forwarders*, pumping out bytes at roughly the same average rate as the receive rate; and some peers act as *sinks*, forwarding very little traffic over their lifetimes.

One important lesson learned is that even though an access link may have asymmetric downstream and upstream bandwidths (such as ADSL), with the downstream bandwidth being higher than the upstream bandwidth, the actual bit rates can have opposite behavior, with uploading rates being higher than the downloading rates. Thus, P2P video streaming can potentially severely stress the upstream capacity of access ISPs.

Note that in trace CCTV10-Residence, the download rate falls significantly below the playback rate for about 4 minutes at about time $t = 33$ minutes. After this decrease, the peer aggressively downloads from the network, downloading at a rate higher than the playback rate for about 3 minutes. Then the download rate becomes steady again. Despite the PPLive and media player buffering, this download rate deficit may have impacted the quality of the video playback.

Although not as high as the two campus peers, the residential peer watching CCTV3 contributed traffic volume comparable to its download traffic volume. However, the other residual peer (watching CCTV10) only uploaded 4.6 Mbytes of video to other peers. Since the two residential peers have similar access bandwidth, we seek an explanation for why this one peer hardly uploaded any video. One possibility is that the other peers contribute sufficient upload bandwidth, so that this residential peer simply doesn't need to contribute. Another possibility is that the buffering and rendering for this residential peer lags behind most of the other peers; thus, relatively few other peers can use the residential peer's chunks (as discussed in Section IV-C).

D. Properties of Video TCP Connections

In this section we examine the basic properties of video TCP connections in PPLive, including connection duration, number of partners, partner churn, and upload/download traffic to/from partners.

1) Duration of Video TCP Connections: A video TCP connection begins with the TCP SYN packet; we say that the connection ends when we see a TCP FIN packet or when we see a packet that is not followed by any other packet in the connection for two minutes. Figure 19 provides a typical Complementary Cumulative Distribution Function (CCDF) of video TCP connection durations. Note that the durations spread over a wide range and have a skewed distribution. The median duration is 22.5 seconds and the mean is 381.1 seconds. Only about 10% of the connections last for over 15 minutes and the longest session lasts for more than 2 hours. Because many connections are short, a peer may only exchange a few video chunks with its partner before the connection ends.

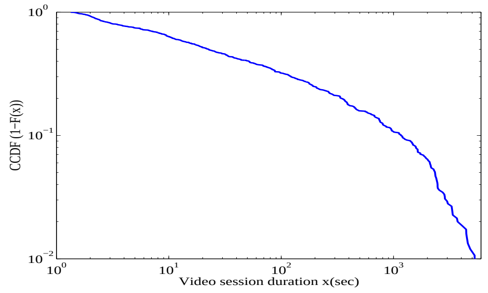
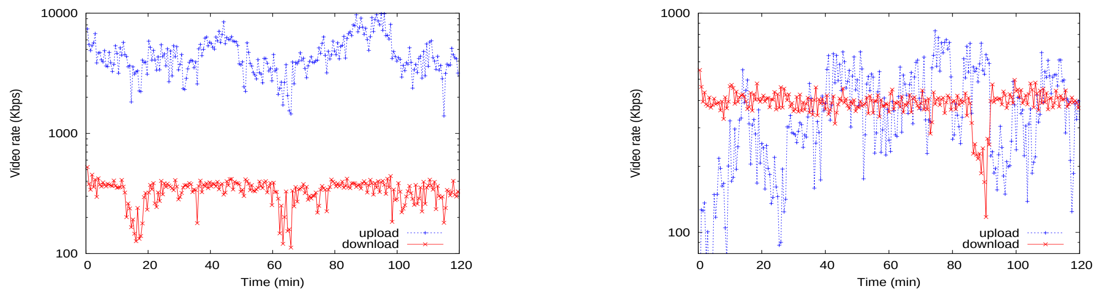
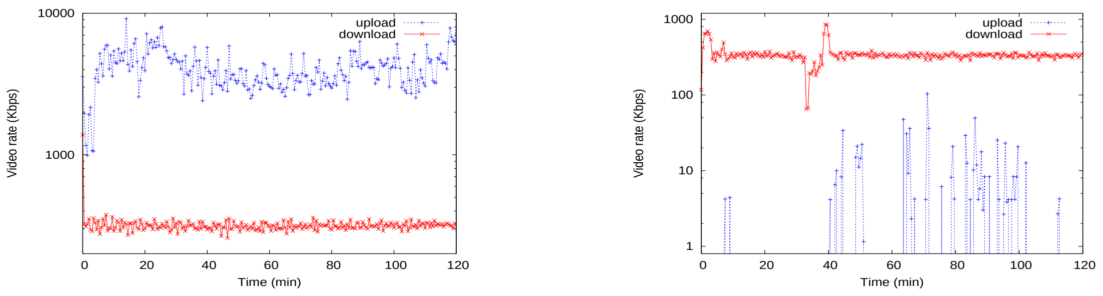


Fig. 19. CCDF of duration of video TCP connection for CCTV3-Campus

2) Number of Partners: For each of the four peers, Figure 20 plots the number of partners (that is, its number of video TCP connections) the peer has as a function of time. Note that campus peers have many more partners than residential peers. A campus peer utilizes its high-bandwidth access, maintaining a steady number of partners for video traffic exchange. Content popularity also has a significant impact on the number of partners for residential peers. In particular, the residential peer with the less-popular CCTV10 channel seems to have difficulty in finding enough partners for streaming the media. At time $t = 33$ minutes, the number of partners drops to 1. This reduction in partners significantly impacts the download



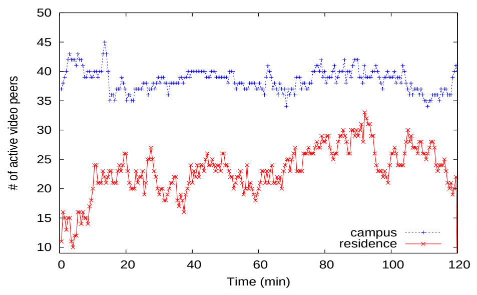
(a) CCTV3-Campus (b) CCTV3-Residence



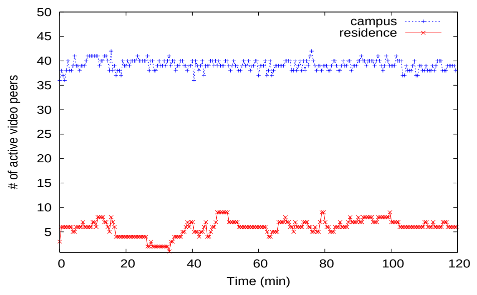
(c) CCTV10-Campus (d) CCTV10-Residence

Fig. 18. Upload and download video bit rates for the four traces

rate of this residential peer, as shown in Figure 18(d). In this experiment, the peer detected this rate reduction quickly and started to search for new partners. New partners were quickly found and fresh streaming flows were established; hence, the video download rate recovered quickly as a result.



(a) CCTV3



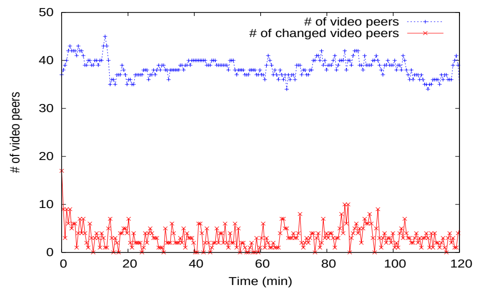
(b) CCTV10

Fig. 20. Evolution of numbers of partners for each of four peers

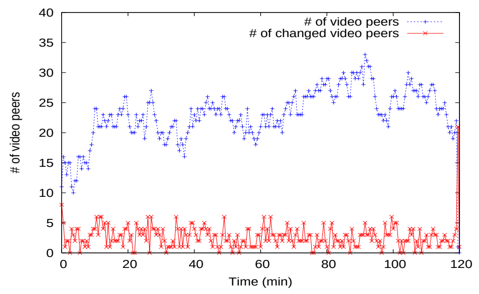
3) Dynamics of Partners: During its lifetime, a peer continually changes its partners. This is illustrated in Figure 21, in which the number of partners (video chunk exchange sessions) is sampled every 30 seconds. A changed partner refers to either a new partner or a partner that stops to exchange video chunks. For both types of access networks, over a 30 second period, typically several partners leave and several new partners arrive. Nevertheless, compared with the total number of partners, the average number of the changed peers in 30 seconds is less than 10% of the total video peers for campus peers. However, the changed partners make up a larger percentage of the total number of partners for residential peers. One consequence is that the download video rates of residential peers are likely to fluctuate more significantly.

4) Locality of Partners: It would be a waste of network resources to download from another continent if a channel could be downloaded from a source in the same continent. We investigated whether a PPLive peer takes locality into account when it determines which peer to download from. We employed the same technique as that of Section III-D to determine the geographic location of a peer.

For the three traced peers (all located in New York) with substantial upload traffic, Table IV shows the geographic distribution of the peer's partners. We observe that a large fraction of partners are located in Asia, and these Asian partners contribute the majority of the download traffic. On the other hand, the majority of the traffic uploaded by each of our traced peers is to partners in North America. For example, in Table IV(b), the CCTV3-residential peer downloads 81.0% video traffic from partners in Asia and 18.3% video traffic from partners in North America; however, it uploads only 6.4% video traffic to Asia and 64.1% to North America. We can see that there are still space of improvement for PPLive to take advantage of peer locality. It is more beneficial for



(a) CCTV3-Campus



(b) CCTV3-Residence

Fig. 21. Partner departures and arrivals

the CCTV3-residential peer to download more video traffic from peers in North America instead of downloading too much video across inter-continental links.

TABLE IV
GEOGRAPHIC DISTRIBUTION OF PARTNERS

(a) CCTV3-Campus

	Asia	North America	Other Places
peer(%)	19.1	73.5	7.4
Download(%)	72.3	26.2	1.5
Upload(%)	1.1	83.9	15.0

(b) CCTV3-Residence

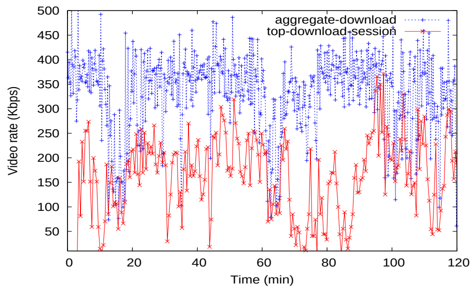
	Asia	North America	Other Places
peer(%)	63.5	29.5	7.0
Download(%)	81.0	18.3	0.7
Upload(%)	6.4	64.1	29.5

(c) CCTV10-Campus

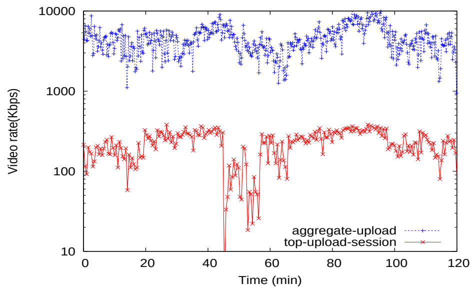
	Asia	North America	Other Places
peer(%)	37.1	55.7	7.2
Download(%)	92.1	6.9	1.0
Upload(%)	2.6	76.2	21.2

5) Traffic Volume Breakdowns across Video TCP Connections: Although PPLive uses a mesh-pull architecture, it is possible that the underlying traffic patterns follow those of a tree, where each peer is primarily fed by one peer over times of minutes. We now disprove this conjecture.

We now examine how the download rates differ among partners and how the upload rates differ among partners. For a campus peer, Figure 22(a) compares the peer's aggregate download video rate with the download rate from the greatest contributing peer. This top peer contributes on average about 50% of the total video download traffic. However, the download rate from this top peer is highly dynamic, most likely due to content availability from the top peer and congestion in the network between the two peers. One important consequence is that a peer typically receives video from more than one peer at any given time. We also plot analogous curves, in log scale, for video upload in Figure 22(b). Since the campus node uploads to many peers, the top peer video upload session only accounts for about 5% of the total video upload traffic.



(a) Download



(b) Upload

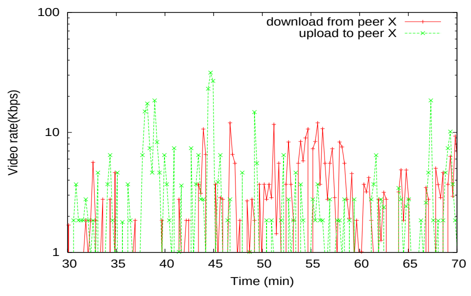
Fig. 22. Peer download and upload video traffic breakdown for CCTV3-Campus

6) Uni-Directional or Bi-Directional Traffic?: Having now shown that a peer can be concurrently fed by multiple parents, one may conjecture that the traffic flows are nevertheless uni-directional, that is, between each pair of partners, traffic flows primarily in one direction over all short time scales. We now show by example that this conjecture does not generally hold. In trace CCTV3-Campus, the campus peer under control has two neighbors A and B. As shown in Figure 23, we plot the video traffic download and upload bit rate between this sniffed peer and its peer neighbors, A and B. The average time interval is 15 seconds. We can clearly observe that there exists bi-directional video traffic exchange between a pair of peers even in a small time scale ($<$ 1 minute).

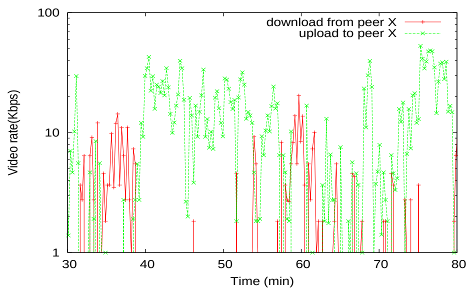
Thus, traffic flows are neither tree-like nor unidirectional, that is, PPLive has a true mesh overlay, without parent/child relationships. These traffic flows are very different from those in a P2P tree-push architecture, but are similar in character to those of BitTorrent. *One important lesson learned from this study is that mesh-pull architectures are more correctly viewed as variations on BitTorrent rather than variations on tree-pull architectures such as end-system multicast.*

VI. Related Work

CoolStreaming [4], the first large-scale mesh-pull live streaming system, demonstrated the feasibility to support a



(a) Peer A



(b) Peer B

Fig. 23. Video traffic exchange between two selected peer neighbors with the sniffed campus peer for CCTV3-Campus

scalable video delivery service with reasonably good viewing quality over the best-effort Internet. Performance evaluation over the Internet and the PlanetLab showed that mesh-pull live streaming systems achieve significant more continuous media playback than tree based systems [20]. Research efforts and prototyping initiatives have continued along this direction from both the academic community and industry [21]. This new paradigm brings opportunities as well as challenges in broadcasting video over the Internet [22]. Researchers have proposed a number of mesh-pull P2P streaming systems [23], [4], [24], [25], [26], [27], [28], [29], [30]. These research systems are usually evaluated via simulation or a small-scale testbed in constrained environments. Recently, there has been a number of theoretical studies of mesh-pull streaming systems [18], [31].

Recently, a number of startup companies have been to provide IPTV services using mesh-pull streaming architectures [5], [7], [8], [9], [10], [11]. The scale of these streaming networks have exceeded that of CoolStreaming significantly. Little is known, however, about these proprietary systems. Researchers have begun to study various performance metrics of these systems, including user behaviors and traffic patterns. The measurement techniques fall into two categories: passive sniffing and active crawling.

For passive sniffing, our previous work in [12] was the first measurement study of a large-scale P2P streaming system to study the traffic pattern and peer dynamics of the PPLive network, which is one of the most popular IPTV systems. Our work was followed by two other passive measurement studies [32] and [33]. Ali et al. [32] focus on the traffic characteristics of controlled PPLive peers on PPLive and SopCast. Passive sniffing was also utilized to study the traffic pattern of PPLive, PPStream, TVAnts and SopCast in [33].

Passive sniffing techniques are often constrained to measure a small set of controlled peers. We also developed active crawling apparatus to measure the global view of the PPLive network in [13]. After our work in [13], another crawler-based measurement study was conducted in [34]. Vu et al. [34] only focused on the measurement of peer dynamics for a small number of PPLive channels.

There are a number of measurement studies of other types of P2P systems, including file sharing, content-distribution, and VoIP. For file sharing, Saroiu et al. measured the Napster and Gnutella [35] and provided a detailed characterization of end-user hosts in these two systems. Their measurement results showed significant heterogeneity and lack of cooperation across peers participating in P2P systems. Gummadi et al. monitored KaZaa traffic [36] for characterizing KaZaa's multimedia workload and they showed locality-aware P2P file-sharing architectures can achieve significant bandwidth savings. Ripeanu et al. crawled the one-tier Gnutella network to extract its overlay topology. For the latest two-tier Gnutella network, Stutzbach et al. provided a detailed characterization of P2P overlay topologies and their dynamics in [37]. Liang et al. deployed active crawling in [38] to reveal in-depth understanding of KaZaa overlay structure and dynamics. In [39], Liang et al. further demonstrated the existence of content pollution and poisoning in KaZaa using an active crawler.

A measurement study was carried out for the live streaming workload from a large content delivery network in [40]. For content distribution, Izal et al. and Pouwelse et al. reported measurement results for BitTorrent [41] and [42]. For VoIP, two measurement studies of Skype are available [43] and [15]. A detailed protocol analysis of Skype was presented in [43] and Skype traffic pattern reported in [15] differs fundamentally from previous file-sharing P2P systems. Performance evaluation of CoolStreaming was carried out over PlanetLab [4] and the measurement results showed that mesh-pull live streaming systems achieve significant more continuous media playback than tree based systems.

VII. Conclusions and Future Work

IPTV is an emerging Internet application which may dramatically reshape the traffic profile in both access and backbone networks. We conducted a measurement study on a popular P2P IPTV application, PPLive. Our study demonstrates that the current Internet infrastructure is capable of providing the performance requirements of IPTV at low cost and with minimal dedicated infrastructure. Through passive and active measurements, we characterized P2P IPTV user behavior and traffic profiles at packet, connection and application levels. More importantly, the measurement results provide an understanding of how to architect a successful large scale P2P IPTV system. Insights obtained in this study should be valuable for the development and deployment of future P2P IPTV systems.

Although large-scale P2P IPTV systems are feasible in today's Internet, this class of applications is in its infancy, and performance remains to be improved in several directions:

- *Shorter Start-up Delay.* We showed that at start-up, PPLive buffers tens of seconds of video before playback to compensate for peer churn and rate fluctuations of video connections. However, many users of ordinary television enjoy rapidly switching among channels. Thus, if P2P IPTV is truly going enhance ordinary television, the start-up delay needs to be reduced to from tens of seconds to a few seconds. Possible directions to be investigated include redundant downloading and/or network coding of video chunks. This would come at the price of increased video traffic redundancy.

- *Higher Rate Streaming.* Unlike BitTorrent file distribution system, it is difficult to enforce the tit-for-tat policy in a P2P streaming system, since many peers have upload capacity less than the compressed playback rate of video. To compensate, peers with higher uploading capacity upload much more than what they download to sustain steady playback at all peers. To support higher bit rates, the workload on those "amplifier" nodes will be further increased. It becomes questionable whether an ordinary peer, and the access ISP to which it is connected, will have the capability and incentive to continue to provide the additional upload traffic. Thus, in the future, some level of dedicated infrastructure (such as dedicated proxy nodes), may have to be combined with the P2P distribution to deliver videos at higher rates.
- *Smaller Peer Lags.* In our measurement study we observed large playback lags, that is, some peers watch frames in a channel minutes behind other peers. To reduce the lags, better peering strategies and video chuck scheduling schemes are needed.
- *Better NAT Traversal.* We observed lots of private IP addresses in collected peer lists. The peers behind NATs are often not fully reachable. To utilize the uploading capacities from peers behind NATs, better NAT traversal schemes need to be employed.

References

[1] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. on Computer Systems*, pp. 85–111, May 1990.

[2] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky, "A transport layer for live streaming in a content delivery network," *Proc. IEEE*, vol. 92, no. 9, pp. 1408– 1419, Sep. 2004.

[3] Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," in *ACM SIGMETRICS*, 2000, pp. 1–12.

[4] 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.

[5] "PPLive," http://www.pplive.com.

[6] B. Cohen, "Incentives Build Robustness in BitTorrent," in *P2P-Econ*, June 2003.

[7] "PPStream," http://www.ppstream.com.

[8] "UUSee." [Online]. Available: http://www.uusee.com/

[9] "SopCast," http://www.sopcast.org/.

[10] "TVAnts," http://www.tvants.com.

[11] "VVSky," http://www.vvsky.com.cn.

[12] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system," in *IPTV workshop in conjunction with WWW2006*, May 2006. [Online]. Available: http://cis.poly.edu/~ross/papers/ppliveWorkshop.pdf

[13] ——, "A measurement study of a large-scale P2P IPTV system," Polytechnic University, Tech. Rep., May 26 2006. [Online]. Available: http://cis.poly.edu/~ross/papers/P2PliveStreamingMeasurement.pdf

[14] "BitTorrent," http://bittorrent.com/.

[15] S. Guha, N. Daswani, and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *IPTPS'06*, Feb. 2006.

[16] "Maxmind," http://www.maxmind.com/app/country.

[17] "Big champagne," http://www.bigchampagne.com/.

[18] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic fluid theory for P2P streaming systems," in *Proceedings of INFOCOM*, 2007.

[19] "Ethereal," http://www.ethereal.com/.

[20] X. Zhang, J. Liu, and B. Li, "On large-scale peer-to-peer live video distribution: Coolstreaming and its preliminary experimental results," in *IEEE MMSP'2005*, Oct. 2005.

[21] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer Internet video broadcast," Nov. 2006.

[22] R. Rejaie, "Anyone can broadcast video over the Internet," *Commun. ACM*, vol. 49, no. 11, pp. 55–57, 2006.

[23] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *IPTPS'05*, Feb. 2005.

[24] M. Zhang, L. Zhao, Y. Tang, J.-G. Luo, and S.-Q. Yang, "Large-scale live media streaming over peer-to-peer networks through global Internet," in *P2PMMS'05*, 2005, pp. 21–28.

[25] C. Dana, D. Li, D. Harrison, and C. N. Chuah, "BASS: BitTorrent assisted streaming system for video-on-demand," in *IEEE MMSP*, Oct. 2005.

[26] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-Peer live streaming," in *IEEE INFOCOM*, Apr. 2006.

[27] F. Pianese, J. Keller, and E. W. Biersack, "PULSE, a flexible P2P live streaming system," in *Global Internet*, Apr. 2006.

[28] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," in *Global Internet*, Apr. 2006.

[29] N. Magharei and R. Rejaie, "Understanding mesh-based peer-to-peer streaming," in *NOSSDAV '06*, May 2006.

[30] T. Piotrowski, S. Banerjee, S. Bhatnagar, S. Ganguly, and R. Izmailov, "Peer-to-peer streaming of stored media: the indirect approach," in *SIGMETRICS '06/Performance '06*, 2006, pp. 371–372.

[31] S. Tewari and L. Kleinrock, "Analytical model for BitTorrent-based live video streaming," in *IEEE NIME 2007 Workshop*, Jan. 2007.

[32] S. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming," in *First Workshop on Recent Advances in Peer-to-Peer Streaming*, Waterloo, Canada, August 10 2006. [Online]. Available: http://wraips.org/measurement.pdf

[33] T. Silverston and O. Fourmaux, "P2P IPTV measurement: A comparison study," University Paris 6 C LIP6/NPA Laboratory, Tech. Rep., Oct. 2006. [Online]. Available: http://www.arxiv.org/abs/cs.NI/0610133

[34] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Mapping the PPLive network: Studying the impacts of media streaming on P2P overlays," Department of Computer Science, University of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS-R-2006-275, Aug. 2006.

[35] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of Napster and Gnutella hosts," *Multimedia Syst.*, vol. 9, no. 2, pp. 170–184, 2003.

[36] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-sharing Workload," in *ACM SOSP*, 2003, pp. 314–329.

[37] D. Stutzbach, R. Rejaie, and S. Sen, "Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems," in *ACM IMC*, Oct. 2005.

[38] J. Liang, R. Kumar, and K. W. Ross, "The FastTrack Overlay: A Measurement Study," *Computer Networks*, vol. 50, no. 6, pp. 842–858, Apr. 2006.

[39] J. Liang, N. Naoumov, and K. Ross, "The Index Poisoning Attack in P2P File-Sharing Systems," in *IEEE INFOCOM*, Apr. 2006.

[40] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the Internet," in *ACM IMC*, 2004, pp. 41–54.

[41] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice, "Dissecting bittorrent: Five months in a torrent's lifetime." in *PAM*, 2004, pp. 1–11.

[42] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The Bittorrent P2P File-sharing System: Measurements and Analysis," in *IPTPS'05*, Feb. 2005.

[43] S. A. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," in *IEEE INFOCOM*, Apr. 2006.