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

Abstract

An emerging Internet application, IPTV, has the potential to flood Internet access and backbone ISPs with massive amounts of new traffic. We recently measured 200,000 IPTV users for a single program, receiving at an aggregate simultaneous rate of 100 gigabits/second. Although many architectures are possible for IPTV video distribution, several chunk-driven P2P architectures have been successfully deployed in the Internet. In order to gain insight into chunk-driven 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 chunk-driven 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 IPTV user behavior, P2P IPTV traffic overhead and redundancy, peer partnership characteristics, P2P IPTV viewing quality, and P2P IPTV design principles.

1 Introduction

With the widespread adoption of broadband residential access, IPTV may be the next disruptive IP communication technology [11]. 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 [14], application-level infrastructure overlays such as those provided by CDN companies [1, 19], peer-to-peer multicast trees such as in end-system multicast [12], and chunk-driven P2P streaming such as CoolStreaming [26] and PPLive [6]. 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.

In terms of the number of simultaneous users, the most successful IPTV deployments to date have employed chunk-driven P2P streaming architectures. Bearing strong similarities to BitTorrent [13], chunk-driven P2P architectures have the following characteristics:

1. A television channel is divided into media chunks (e.g., each chunk consisting of one second of media data) and is made available from an origin server.

2. A host, interested in viewing a particular channel, requests from the system a list of hosts currently watching the channel. The host then establishes partner relationships (TCP connections) with a subset of hosts on the list.

3. Each host viewing the channel buffers and shares chunks with other hosts viewing the same channel. In particular, a host periodically receives buffer maps from each of its current partners. The buffer map indicates the chunks the partner currently has available. Using a scheduling algorithm, the host requests from its partners the chunks that it will need in the near future.

4. As in BitTorrent, each host continually seeks new partners that deliver data at higher rates than its existing partners.

An important characteristic of chunk-driven P2P algorithms is the lack of an (application-level) multicast tree - a characteristic particularly desirable for the highly dynamic, high-churn P2P environment [26]. Although these chunk-driven algorithms have similarities with BitTorrent, BitTorrent in itself is not a feasible delivery architecture, since it does not account for the real-time needs of IPTV.

Several chunk-driven P2P streaming systems have been successfully deployed to date, accommodating 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 chunk-driven 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 [6], ppStream [7], VVSky [9], TVAnts [8] and FeiDian [4].

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 chunk-driven 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 [17]. 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?
- *How good is the viewing quality?* What are the viewing quality metrics in an IPTV system? How well does PPLive perform with respect to these metrics?
- *What are the fundamental requirements for a successful chunk-driven 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?

In this paper, 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.

This paper is organized as follows. We conclude this section with an overview of related P2P measurement work. In Section 2, we provide an overview of different aspects of PPLive including architecture, signal and management protocols based on our measurement studies. The design and development of the tools are presented in details in Section 3. Our measurement tools include an active crawler and a passive sniffer. In Section 4, 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. In Section 5, by sniffing monitored peers, we present the traffic patterns and peering strategies as viewed by residential and campus PPLive clients. In Section 6, we characterize the streaming performance of PPLive, including playback freezing and restoration, using our playback monitor. Finally, based on our measurement results, we outline some design guidelines for the successful deployment of IPTV application over the Internet in Section 7.

1.1 Related P2P Measurement Work

To our knowledge, this paper (along with [17]) is the first measurement study of a large-scale P2P streaming system. There are, however, a number of recent 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 [23] 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 [16] 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 [25]. Liang et al. deployed active crawling in [20] to reveal in-depth understanding of KaZaa overlay structure and dynamics. In [21], 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 [24]. For content distribution, Izal et al. and Pouwelse et al. reported measurement results for BitTorrent [18] and [22]. For VoIP, two measurement studies of Skype are available [10] and [15]. A detailed protocol analysis of Skype was presented in [10] and Skype traffic pattern reported in [15] differs fundamentally from previous file-sharing P2P systems. Performance evaluation of CoolStreaming was carried out over PlanetLab [26] and the measurement results showed that chunk-driven live streaming systems achieve significant more continuous media playback than tree based systems.

2 Overview of PPLive

PPLive is a free P2P IPTV application. According to the PPLive web site [6] 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 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. The PPLive web site [6] provides limited information about its proprietary technology. Through our measurement studies and protocol analysis, however, we have gained significant insight into the PPLive protocols and streaming mechanisms. In order to gain a better understanding of our measurement tools and results, in this section we provide an overview of the PPLive operation. The overview also provides an introduction into the design of a chunk-based video streaming system.

The PPLive software, running in user computers (peers), has two major communication protocols: (i) a registration and peer discovery protocol; and (ii) a P2P chunk distribution protocol. Figure 1 depicts an overview of the registration and peer discovery protocol. When an end-user starts the PPLive software, it joins the PPLive network and becomes a PPLive peer node. The first action (step 1) is an HTTP exchange with the PPLive Web site to retrieve a list of channels distributed by PPLive. Once the user selects a channel, the peer node registers with the bootstrap root servers and requests a list of peers that are currently watching the channel (step 2). The peer node then communicates with the peers in the list to obtain additional lists (step 3), which it aggregates with its existing 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 service port numbers. The registration and peer discovery protocol is commonly running over UDP; however, if UDP fails (for example, because of a firewall), PPLive will instead use TCP for registration and peer discovery.

We now describe the chunk distribution protocol. At any

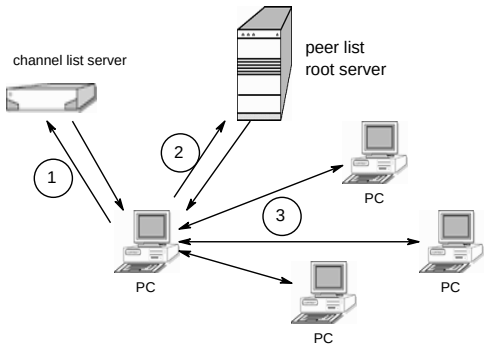


Figure 1: Channel and peer discovery

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). The offset field is of 4 bytes. For one PPLive 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. Figure 2 illustrates a buffer map. A peer

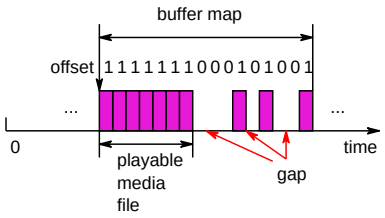


Figure 2: A peer's buffer map of video chunks

can request, over a TCP connection, a buffer map from any peer in its current list of peers. After a peer A receives a buffer map from peer B, peer A can request one or more chunks that peer B has advertised in the buffer map. As we will see in Section 5, a peer A may download chunks from tens of other peers simultaneously. PPLive continually searches for new partners from which it can download chunks. Since PPLive is proprietary, we do not know the exact algorithm a peer uses for choosing partners and requesting chunks. 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 [2] [26]). Peers can also download chunks from the PPLive channel server. The chunks are sent over TCP connections.

Having addressed how chunks are distributed among peers, we now briefly describe the video display mecha-

nism. As mentioned above, PPLive works in conjunction with a media player (either Windows Media Player or RealPlayer). Figure 3 illustrates the interaction between the PPLive peer software and the media player. The PPLive engine, once having buffered a certain amount of contiguous chunks, launches the media player. The media player then makes an HTTP request to the PPLive engine, and the PPLive 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.

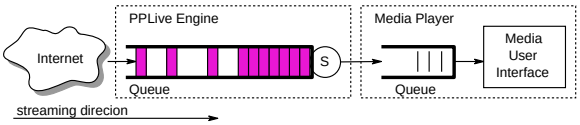


Figure 3: PPLive streaming process

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

3 Measurement Methodologies

Our P2P network measurements fall into two categories: active crawling and passive sniffing. The active crawling is used to obtain a global view of the entire PPLive network for any channel. Our crawler has a UDP component for collecting peer lists from all the PPLive peers; and a TCP component for collecting buffer maps from all the PPLive peers. The passive sniffing is used to gain a deeper insight into PPLive from the perspective of residential users and campus users.

3.1 Active Crawling

To characterize the behavior of the entire PPLive network, we developed a crawler which repeatedly probes *all* of the PPLive nodes that are watching a specific channel. Building the crawler was a major challenge in itself, since we needed to implement portions of the PPLive proprietary protocol. To this end, using packet traces from passive sniffing and our knowledge about how chunk-driven P2P streaming generally operates, we were able to understand critical portions of the PPLive's signaling protocols. During a crawling experiment, peer responses are recorded for online processing and offline analysis.

3.1.1 Harvesting Peer Lists

Recall from Section 2 that each peer watching a particular channel maintains a peer list, which lists other peers currently watching the same channel. Also recall that any peer A can send to any other peer B, within a UDP datagram, a request for peer B's peer list. The crawler, implementing the PPLive protocol, sweeps across the peers watching the channel and obtains the peer list from each visited peer. The crawler does these three phases:

- *Peer Registration:* The UDP crawler first registers itself with one of the root servers by sending out a peer registration packet. The significant information in this packet 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 generates a new, random value every time it re-joins the channel.
- *Bootstrap:* After the registration, the crawler sends out multiple bootstrap peer list queries to the peer-list root server for peers enrolled in this channel. In response to a single query, the server will return a list of peers (normally 50 peers), including IP addresses and service port numbers. The crawler aggregates all the lists it has received, thereby maintaining its own list of peers enrolled in the channel.
- *Peer Query:* After obtaining an initial peer list from the root servers, the crawler sends peer list queries directly to those peers from the initial list. Active peers will return part of their own peer lists, which get added to the crawler's list.

PPLive clients are highly dynamic, joining and leaving PPLive and switching between channels frequently. To account for the highly dynamic nature of the PPLive clients, we designed the crawler to periodically crawl the PPLive network to track active peers once a minute. In particular, every one minute, the crawler starts from scratch, with an empty peer list. Furthermore, within each minute, the crawler is only active for the first 15-seconds, during which it:

1. Obtains an initial peer list for the root server.
2. Sends peer queries to non-queried peers in the list.
3. Marks a peer active if it responds to the query with its own peer list; expands the crawler's peer list using the lists received from active peers; returns to step (2) until no new peers are obtained by peer queries.

Then we have finished one *loop* of probing. In our experiments, it normally takes about 6 seconds to finish one loop. After one loop, to include peers that joined the network during the probing loop, the crawler goes back to the beginning of the list and probes all the peers again to find new active peers. We repeat the process until 15 seconds run up. The crawler then records the active peers seen in this 15 seconds, clears its memory, and goes to sleep until the beginning of

the next minute. In this manner we obtain a profile of the active peers every minute. We note in passing that the crawler doesn't use a NAT traversal scheme. Peers behind NATs may not set their NATs properly to receive peer queries from the crawler. Our crawler therefore under-estimates the number of active peers; in fact, in some of experiments we observed that as many as 37% of the PPLive peers could be behind NATs. Although NATs make it difficult to determine the absolute number of users at any time, our measurement results still enable us to report time evolutionary trends and also provide lower bounds on the number of users.

3.1.2 Harvesting Buffer Maps from Active Peers

To monitor the buffer content of PPLive clients, we augmented the crawler with a TCP component that retrieves the buffer map from the active peers during above crawling process. To this end, 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 the remote peers throughout the network.

3.2 Passive Sniffing

Passive sniffing captures the traffic exchanged between our monitored peers and their partners in the PPLive network. We 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 in New York. Each PC ran Ethereal [3] 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.

3.2.1 Playback Monitoring

In IPTV, user's perceived quality is vital for a successful service. As shown in Figure 3, the media player interacts with the PPLive engine. Whenever the PPLive engine has received playable media chunks, the PPLive engine streams these media chunks to the player. When the media player runs out of media contents, the player freezes, impacting the user perceived quality. To trace the user playback performance, we developed a simple PPLive playback monitor. This monitor emulates the normal media playback process and tracks the presentation time of the latest media chunk. The difference between the actual playback time and the latest media chunk presentation time indicates the size of playable media size in the player. The monitor reports an playback freeze whenever this time difference reaches 0. During the playback freezing period, the monitor continues receiving media content from the PPLive engine. When the buffered content is above a threshold, which is specified by the content source in the media file header, the monitor reports a recovery from freeze.

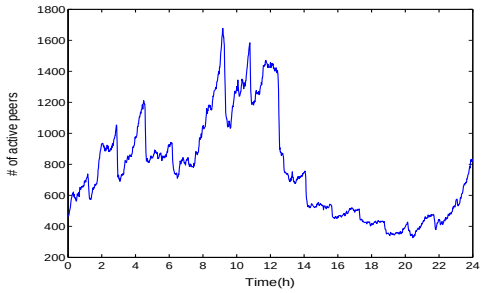
4 Global Behavior of the PPLive Network

In this section, we report *global statistics* for the PPLive network collected by UDP component of the crawler. PPLive hosts more than 200 different programs. To compare the characteristics of different programs, we crawled two programs: XING, a popular channel with a 5-star popularity grade: and GUANG, a less popular channel with a 1-star popularity grade. The two programs were both crawled for one entire day in April, 2006. The crawler crawled both programs every minute as described in the previous section. Based on the lists of the active peers at every minute of the crawled programs, we calculated the number of users, user arrivals and departures, and user geographic locations. What'more, we also collected data during the Chinese New Year to observe some user behavior.

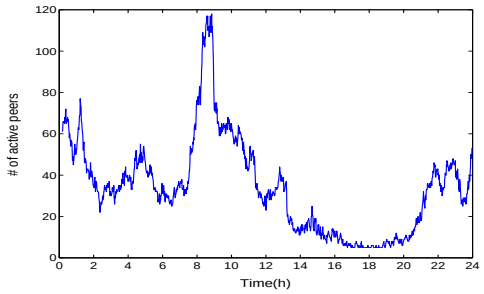
4.1 Number of Participating Users

Figure 4 shows how the number of users evolves for both crawled programs. Both sub-figures are labelled in US Eastern Standard Time. We first observe that the numbers of participating users are quite different for the two programs. The maximum number of users for the popular program reaches nearly $1,700$; however, that of the unpopular program is just around 120. The diurnal trend is clear for both programs. The major peaks appear during 9AM to 1PM EST, translating into 9PM to 1AM China local time. As we shall see, those peaks are mostly contributed by users from China. There are several smaller peaks scattered around 9PM to 5AM, translating into 9PM EST to 2AM WST. We believe those are mostly due to users in US. We will show the user geographic distribution in Section 4.3. 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.

In Fig 5, we plot the evolution of the number of users for the popular channel over one week. We can observe that more people use PPLive during weekends than during weekdays. This again confirms that most users use IPTV in their leisure time. 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 rebroadcasting the event. Figure 6 plots



(a) Popular Channel



(b) Less Popular Channel

Figure 4: Diurnal Trend of Number of Participating Users

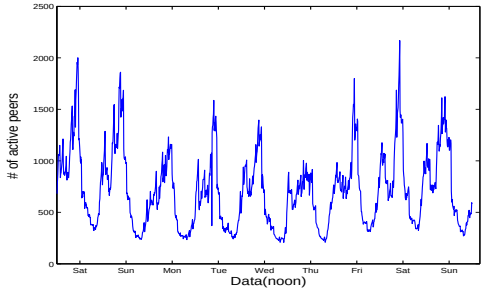


Figure 5: Weekly Trend of Number of Users

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 stayed at this high level for around 4 hours. The number of users went back to normal when the event finished at about 12AM EST. The near constant user population during the event suggests that chunk-driven P2P streaming systems scales well, handling a flash crowd in a live broadcasting. We will come back to the video quality issue in Section 6.

4.2 User Arrivals and Departures

In this section, we examine the peer arrival and departure pattern for the popular channel. We record a peer arrival when the crawler finds a new IP address joining the program. To deal with possible losses of peer queries and responses, we only record a peer departure if a previously active peer doesn't respond to three consecutive queries.

The numbers of peer arrivals and departures of the popular channel in every minute of one day are plotted in Figure 7. Since it is a popular program, many peers continuously

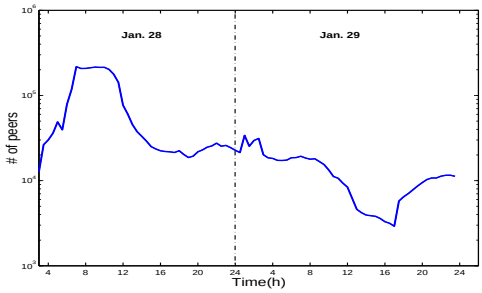
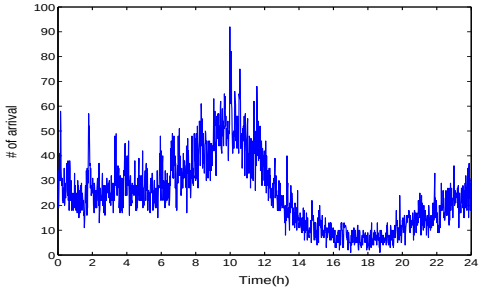
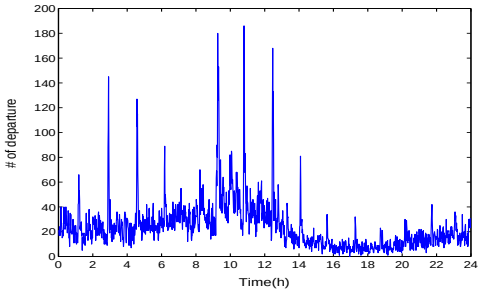


Figure 6: Flash Crowd on Chinese New Year Eve

join and leave. Comparing this figure with that the evolution of the number of active peers, Figure 4(a), we see peers join and leave at a higher rate at the peak time. Note the consecutive pulse spacings of about 2 hours in Figure 7(b). The pulses are due to many peers leaving immediately and simultaneously at the end of (roughly) two-hour programs. This batch-departures pattern in PPLive system 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 more in Section 5.2.3.



(a) Peer Arrival Rate



(b) Peer Departure Rate

Figure 7: Peer Arrival and Departure Evolution of a Popular Channel

We define the peer lifetime 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 8 suggests that the 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.

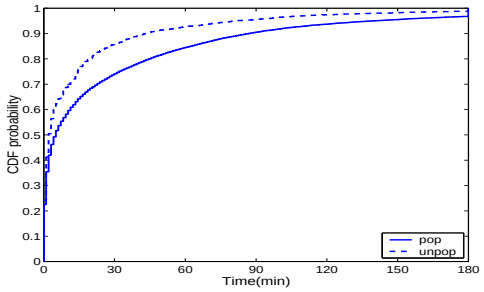


Figure 8: Peer Lifetime Distribution

4.3 User Geographic Distribution

We classify PPLive users into three categories according to their IP addresses: users from Asia, users from North America, and users from the rest of the world. To accomplish this, we query the free MaxMind GeoIP database [5] to obtain the country that each user belongs to with the recorded user IP address as the input. Figure 9 shows the evolution of the geographic distribution of the popular channel during one entire 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 curve reaches it lowest point around 7PM to 8PM EST.

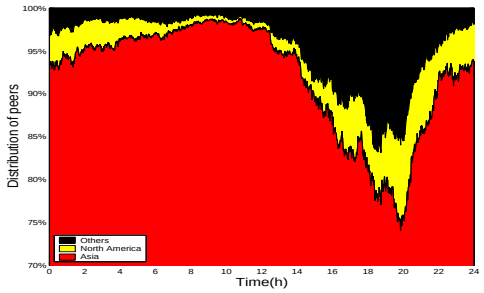


Figure 9: Evolution of Geographic Distribution

Fig 10 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 9, with three regions denoting three different geographical regions. We can see that for this event, many peers from outside of Asia were watching the live broadcast – in fact, a significantly higher percentage of peers were from outside of Asia as compared with Figure 9. The distribution evolution is consistent with the observations in Section 4.2: 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.

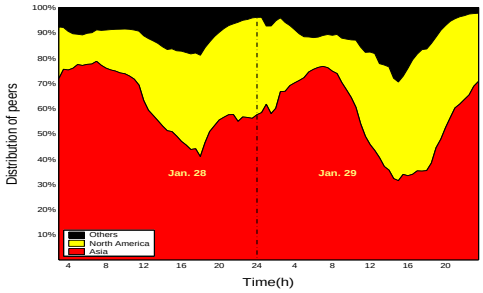


Figure 10: Evolution of Geographic Distribution during Chinese New Year's Eve

5 Local Views from Monitored Peers

In this section, we present detailed statistics derived from packet traces collected on our monitored peers. As summarized in Table 1, 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 was 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.

5.1 Video Traffic and Signaling Overhead

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 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 chunk-driven 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 signalling 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 bitrate). 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

Table 1: Data sets

Trace Name	Trace size (Byte)	Duration (Sec)	Playback Rate (Kbps)	Total IPs	Active IPs	Download (MByte)	Upload (MByte)
CCTV3-Campus	784,411,647	7676	340	3105	2691	360.99	4574.57
CCTV3-Residence	132,494,879	7257	340	1616	1183	372.53	352.75
CCTV10-Campus	652,000,813	7285	312	1008	910	317.08	3815.34
CCTV10-Residence	66,496,909	9216	312	797	282	385.50	7.68

a "video TCP connection"; otherwise, the connection is labeled as a "signaling TCP connection". We filter from the traces all signaling TCP connections.

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 2 we see that the signaling overhead is generally in the 5% to 8% range.

5.1.1 Video Traffic Redundancy

Due to the distributed nature of chunk-driven 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 redundancy traffic once 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 5.1, 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 3, we observe that the traffic redundancy to be 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 11(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.

5.1.2 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 11 plots the upload and download rates for 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 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

Table 2: 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 3: 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%

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 (we will discuss this lagging issue in more detail in Section 6).

5.2 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.

5.2.1 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 12 provides a typical Complementary Cumulative Distribution Function (CCDF) of video TCP connection durations. Note that durations spread over a wide range. The median duration is about 20 seconds and only about 10% of the connections last for over 15 minutes. Because many connections are short, a peer may only exchange a few video chunks with its partner before the connection ends.

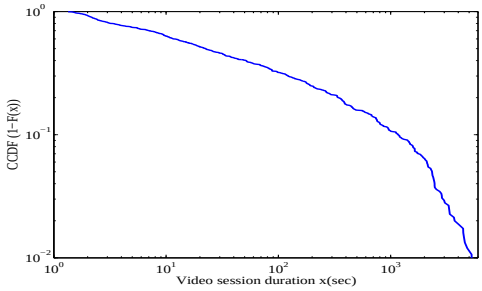
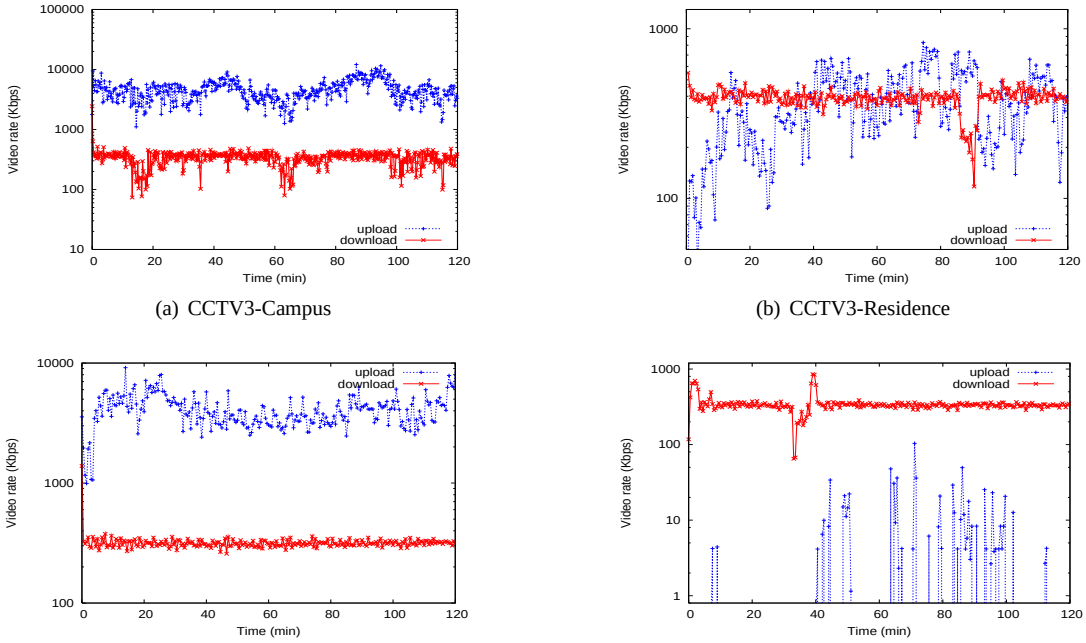


Figure 12: CCDF of duration of video TCP connection for CCTV3-Campus

5.2.2 Number of Partners

For each of the four peers, Figure 13 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 the residential peer. 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 rate of this residential peer, as shown in Figure 11(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-Campus

(b) CCTV3-Residence

(c) CCTV10-Campus

(d) CCTV10-Residence

Figure 11: Upload and download video bit rates for the four traces

5.2.3 Dynamics of Partners

During its lifetime, a peer continually changes its partners. This is illustrated in Figure 14, 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.

5.2.4 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 4.3 to determine the geographic location of a peer.

For the three traced peers (all located in New York) with substantial upload traffic, Table 4 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 4(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.

Table 4: Geographic distribution of partners

(a) CCTV3-Campus

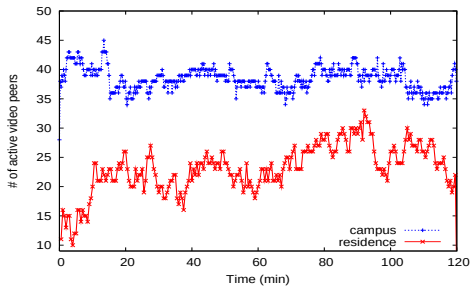
	Asia	North America	Other Places
peer(%)	19.1	73.5	7.4
Download(%)	72.3	26.2	1.5
Upload(%)	1.1	83.9	15.0

(b) CCTV3-Residence

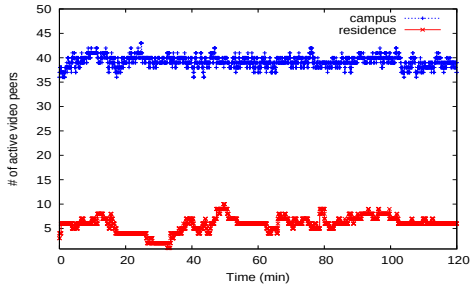
	Asia	North America	Other Places
peer(%)	63.5	29.5	7.0
Download(%)	81.0	18.3	0.7
Upload(%)	6.4	64.1	29.5

(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

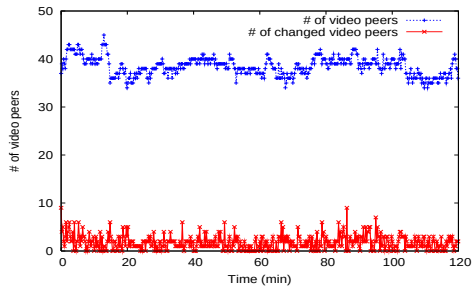


(a) CCTV3

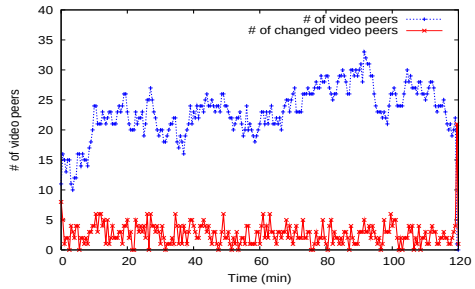


(b) CCTV10

Figure 13: Evolution of numbers of partners for each of four peers



(a) CCTV3-Campus



(b) CCTV3-Residence

Figure 14: Partner departures and arrivals

5.2.5 Traffic Volume Breakdowns across Video TCP Connections

In a chunk-driven P2P system, a peer may download from many partners and may upload to many partners. In this subsection, we examine how the download rates differ among partners and how the upload rates differ among partners. For a campus peer, Figure 15(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. With this multi-download feature, the aggregate video download rate becomes quite smooth. In conjunction with the buffering mechanisms, PPLive typically provides smooth video playback, as discussed in Section 6.3. We also plot analogous curves, in log scale, for video upload in Figure 15(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.

6 User Perceived Quality

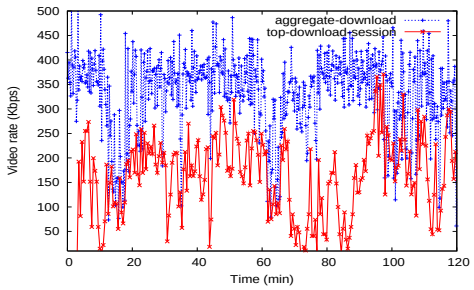
A fundamental measure of the success of an IPTV system is user viewing satisfaction. While both subjective and anecdotal feedback from PPLive users and the stableness of user population suggest highly satisfactory user perceived quality, in this section, we report *quantitative* results on user perceived quality for PPLive. In particular, we used our measurement platforms, to obtain insights into start-up delay, playback freezing rate, and video buffer occupancy.

6.1 Start-up Delay

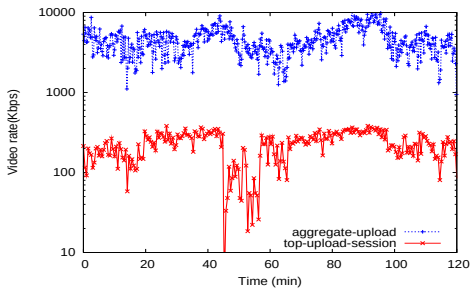
Start-up delay is the time interval from when one channel is selected until actual playback starts on the screen. For streaming applications in best-effort networks, start-up buffering has always been a useful mechanism to deal with the rate variations of streaming sessions. P2P streaming applications additionally have to deal with peer churns, 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. Using our monitored peers, 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 player pop-up delay was in general 10 to 15 seconds and the player buffering delay was 10 to 15 seconds. Therefore, the total start-up delay is from 20 to 30 seconds. 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.

6.2 Playback Continuity

Other than start-up delay, playback continuity is another important measure of user perceived quality. We developed a simple PPLive playback monitor to trace the play-



(a) Download



(b) Upload

Figure 15: Peer download and upload video traffic breakdown for CCTV3-Campus

back performance, as described in Section 3.2.1. We used this plaback monitor to investigate playback continuity for a number of different channels at different times at different locations. Figure 16 shows the player buffer level a CCTV news channel playing back at a campus node. We see that in the time interval [16, 23] minutes, the monitor detected 4 playback freezing incidents. Nevertheless, playback restarted shortly afterward each incident. Interestingly, the player buffer level increases after each recovery. Thus it appears that PPLive adaptively builds buffer levels as a function of streaming quality degradation. Table 5 reports the playback statistics of various traces. The playback freeze probability is approximated by the ratio of the average freezing time interval to the sum of the average freezing and average continuous interval. We observe that the fraction of time that the player freezes is very small for all but one of the traces. When freezing occurs, the average freeze interval of one channel exceeds 1 minute.

6.3 Video Buffering

In the previous section we studied video buffering levels at specific peers by directly monitoring the buffer levels at the peers with our custom-made player monitor. Although these insights provide anecdotal insight into perceived quality, it is desirable to quantify buffer levels across *all* peers actively watching a specific channel. To achieve this goal, we need a mechanism to infer the buffer levels on all active remote peers. To this end we augmented our PPLive crawler to not only harvest peer lists but also buffer maps. From the harvested buffer maps, we can estimate the distribution of buffer

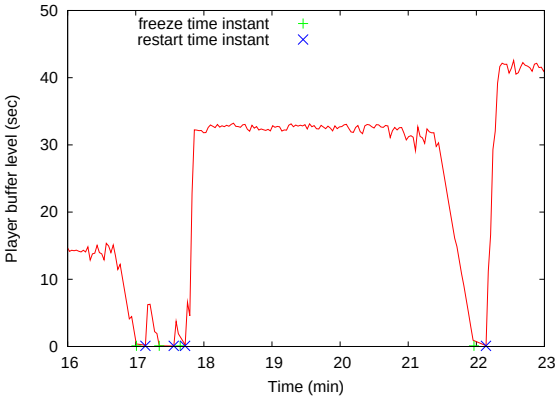


Figure 16: PPLive playback, freeze and re-start of the CCTV news channel for a peer with LAN access

levels across all active peers in a channel.

Figure 17 depicts 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. From the figure, we see that peers

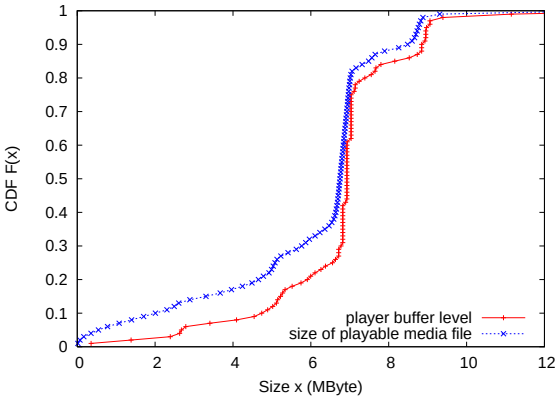


Figure 17: CDF of buffer level for a game channel among 200+ peers over a 600-minute crawling period.

seem to strive to buffer levels of 7 MBytes or higher. With only a small probability peer buffers went under 1 MByte. This again confirms our playback continuity results reported by our playback monitor for local peers in Section 6.2 .

6.4 Playback Lags among Peers

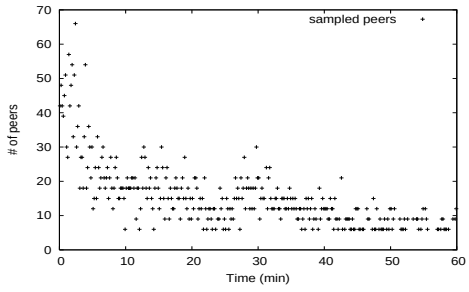
One unfortunate characterisitc of a chunk-driven P2P streaming system is the possibility of playback lags among peers. Specifically. some peers watch frames in a channel minutes behind other peers. Thus, for example, in a soccer game some peers will see a goal minutes after other peers. Aditionally, 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 the 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

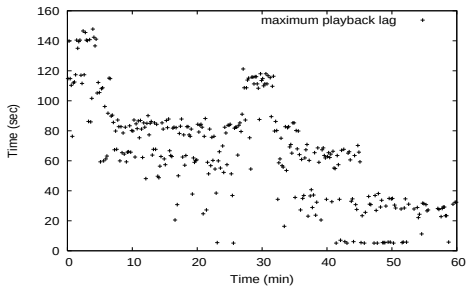
Table 5: Playback Statistics

Trace	Average smooth playback interval	Average freeze interval	Freezing probability
ATV-Campus-NY	4222.6	6.8	0.0016
ATV-Campus-HK	19360.3	61.5	0.0032
ATV-Cable-NY	14789.2	16.9	0.0011
CCTV3-Campus-HK	1462.7	2.4	0.0016
CCTV3-Cable-NY	973.0	2.0	0.0021
CCTV6-Campus-HK	587.2	4.6	0.0078
CCTVNews-Campus-NY	323.8	8.9	0.0268
CCTVNews-Cable-NY	7765.7	6.4	0.0008

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 in a TV channel, ATV, to retrieve their buffer maps for 60 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 10 seconds. For buffer maps within each bin, we calculated the difference between the maximum and minimum offset. Figure 18(a) plots the numbers of buffer maps falling into different time bins, and Figure 18(b) plots the maximum playback time differences over all bins. We observe that the lag among peers can be huge - with around 70 probed peers, the maximum playback time difference of these peers are around 150 seconds.



(a) Sampling Intensity



(b) Maximum playback time difference

Figure 18: Bitmap harvesting for the ATV channel over a 60-minute period.

7 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 will 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 televison enjoy raplidly switching among channels. Thus, if P2P IPTV is truly going to mimic (and 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. It will come at the price of increased video traffic redundancy.
- *Higher Rate Streaming.* We demonstrated that, unlike the 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 pro-

vide 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] Akamai. `http://www.akamai.com/`.

[2] BitTorrent. `http://bittorrent.com/`.

[3] Ethereal. `http://www.ethereal.com/`.

[4] FeiDian. `http://tv.net9.org`.

[5] Maxmind. `http://www.maxmind.com/app/country`.

[6] PPLive. `http://www.pplive.com`.

[7] ppStream. `http://www.ppstream.com`.

[8] TVAnts. `http://www.tvants.com`.

[9] VVSky. `http://www.vvsky.com.cn`.

[10] BASET, S. A., AND SCHULZRINNE, H. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *IEEE INFOCOM* (Apr. 2006).

[11] CHERRY, S. The Battle for Broadband. *IEEE Spectrum 42*, 1 (Jan. 2005), 24 – 29.

[12] CHU, Y., RAO, S. G., AND ZHANG, H. A Case for End System Multicast. In *ACM SIGMETRICS* (2000), pp. 1–12.

[13] COHEN, B. Incentives Build Robustness in BitTorrent. In *P2P-Econ* (June 2003).

[14] DEERING, S., AND CHERITON, D. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Trans. on Computer Systems* (May 1990), 85–111.

[15] GUHA, S., DASWANI, N., AND JAIN, R. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *IPTPS'06* (Feb. 2006).

[16] GUMMADI, K. P., DUNN, R. J., SAROIU, S., GRIBBLE, S. D., LEVY, H. M., AND ZAHORJAN, J. Measurement, Modeling, and Analysis of a Peer-to-Peer File-sharing Workload. In *ACM SOSP* (2003), pp. 314–329.

[17] HEI, X., LIANG, C., LIANG, J., LIU, Y., AND ROSS, K. W. Insights into PPLive: A measurement study of a large-scale P2P IPTV system. In *IPTV workshop in conjunction with WWW2006* (May 2006).

[18] IZAL, M., URVOY-KELLER, G., BIERSACK, E. W., FELBER, P., HAMRA, A. A., AND GARCÉS-ERICE, L. Dissecting bittorrent: Five months in a torrent's lifetime. In *PAM* (2004), pp. 1–11.

[19] KONTOTHANASSIS, L., SITARAMAN, R., WEIN, J., HONG, D., KLEINBERG, R., MANCUSO, B., SHAW, D., AND STODOLSKY, D. A transport layer for live streaming in a content delivery network. *Proc. IEEE 92*, 9 (Sep. 2004), 1408– 1419.

[20] LIANG, J., KUMAR, R., AND ROSS, K. W. The FastTrack Overlay: A Measurement Study. *Computer Networks 50*, 6 (Apr. 2006), 842–858.

[21] LIANG, J., NAOUMOV, N., AND ROSS, K. The Index Poisoning Attack in P2P File-Sharing Systems. In *IEEE INFOCOM* (Apr. 2006).

[22] POUWELSE, J., GARBACKI, P., EPEMA, D., AND SIPS, H. The Bittorrent P2P File-sharing System: Measurements and Analysis. In *IPTPS'05* (Feb. 2005).

[23] SAROIU, S., GUMMADI, K. P., AND GRIBBLE, S. D. Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts. *Multimedia Syst. 9*, 2 (2003), 170–184.

[24] SRIPANIDKULCHAI, K., MAGGS, B., AND ZHANG, H. An analysis of live streaming workloads on the internet. In *ACM IMC* (2004), pp. 41–54.

[25] STUTZBACH, D., REJAIE, R., AND SEN, S. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems. In *ACM IMC* (Oct. 2005).

[26] ZHANG, X., LIU, J., LI, B., AND YUM, T.-S. P. DONet/CoolStreaming: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In *IEEE INFOCOM* (Mar. 2005), vol. 3, pp. 2102 – 2111.