Understanding the Start-up Delay of Mesh-pull Peer-to-Peer Live Streaming Systems

Xiaojun Hei[†], Yong Liu[‡] and Keith W. Ross[†]

[†]Department of Computer and Information Science

[‡]Department of Electrical and Computer Engineering

Polytechnic Institute of New York University

Brooklyn, NY, USA 11201

heixj@poly.edu, yongliu@poly.edu and ross@poly.edu

October 31, 2008

Abstract

An emerging Internet application, peer-to-peer (P2P) video streaming, would revolutionize the entertainment and media industries. In this paper, we have conducted an in-depth measurement study on the start-up delay performance of one of the most popular P2P streaming systems, namely, PPLive. We developed a dedicated PPLive traffic sniffer, which enables us to dissect the start-up delay performance of the PPLive streaming system. Our measurement results show long start-up delays of the PPLive system. We also correlate various factors with the measured start-up delay components in order to reveal the major delay bottleneck. This delay dissection analysis leads to the major reason due to the slow speed in finding "good" video peers the local peer to download video content. Insights obtained in this study will be valuable for the development and deployment of future P2P streaming systems with low start-up delay.

Keyword: Measurement, Peer-to-peer streaming, Delay

1 Introduction

Video streaming may be the next disruptive Internet application with the quick penetration of broadband residential access and the technology progress in video compression. The emerging peer-to-peer (P2P) networks have appeared to be the most promising driving wheel for the large scale video streaming [1–3]. P2P streaming networks do not rely on a dedicated delivery infrastructure and hence offer the possibility of a rapid

deployment at low cost. P2P video streaming would revolutionize the entertainment and media industries. There exist two major design issues for constructing a P2P streaming network: (i) how to form an overlay topology between peers; (ii) how to deliver video content efficiently. The current approaches can be classified into two major categories: (i) peers form a tree-shaped overlay and video content is pushed from the origin server to peers, namely, *tree-push*; (ii) peers form a mesh-shaped overlay and they pull video from each other for content delivery, namely, *mesh-pull*. Over years, many tree-push systems have been proposed and evaluated in academia and achieved some successes. However, they have never taken off commercially. In the past several years, mesh-pull streaming systems (also known as data-driven or BitTorrent-like) have shown major advantages, including system simplicity and inherent robustness, particularly desirable for highly dynamic, high-churn P2P environment. These mesh-pull architectures have enjoyed many successful large scale deployments including CoolStreaming [2], PPLive [4], PPStream [5], UUSee [6], SopCast [7], TVAnt [8], and many others.

The users' viewing experience in streaming applications is crucial for a successful service deployment. Important video quality metrics in P2P streaming include video playback continuity, start-up delay, channel switching latency, playback time lags among users, and so on. Start-up delay is the time interval from when one channel is selected until actual playback starts on the screen. For streaming applications on 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. While short start-up delay is desirable, certain amount of start-up delay is necessary for continuous playback. Our previous measurement study has shown that mesh-pull streaming systems are able to achieve satisfactory download rate to maintain smooth video playback at 300 kbps up to 800 kbps; however, these systems often incur significant long start-up delays, up to a few minutes [9, 10].

The delay performance has become one of the major research issues in improving user's quality-of-experience (QoE) of video streaming applications [11–14]. In this measurement study, we dissect the start-up delay components in mesh-pull streaming systems. The understanding of the start-up delay bottleneck of mesh-pull systems may provide insights into reducing the long delays effectively. In addition, the start-up delay in mesh-pull streaming systems is closely related to the channel switching delay in these systems. The effective delay minimization schemes may also be of good reference on decreasing the channel switching delay in mesh-pull systems.

This rest of the paper is organized as follows. In Section 2, we provide an overview of mesh-pull P2P streaming systems. Based on our understanding on the streaming architecture and the protocol design, we propose the dissecting heuristics on the start-up delay of mesh-pull streaming systems in Section 3. In Section 4, we present

our measurement methodologies for dissecting the start-up delay performance of mesh-pull P2P streaming systems and validate our delay dissection method using detailed trace analysis. In Section 5, we report the measurement results of the start-up delay performance in PPLive. In Section 6, we correlate various factors with the start-up delay of PPLive and reveal the reasons of the incurred long start-up delay. In Section 7, we provide an overview on various promising approaches on reducing start-up delay of P2P streaming systems. Finally, we summarize the paper in Section 8 and outline a few directions of future work.

2 Mesh-pull P2P Streaming Architecture

A number of mesh-pull streaming systems have been deployed; however, most mesh-pull streaming systems provide little information about their proprietary technologies. Through our measurement study and protocol analysis on a few mesh-pull streaming systems [9, 10], we have gained significant insights into the protocols and streaming mechanisms of mesh-pull streaming systems. In spite of different features and implementation details, they share a common generic architecture, which we describe in this section.

2.1 System Overview

As shown in Figure 1, mesh-pull P2P architectures have the following characteristics. A video is divided into media chunks and is made available from an origin server for broadcast. All the video information is accessible for users at the channel server. A host, interested in viewing the video, requests from the channel server the available video streams (step 1 in Figure 1). The tracker server maintains the list of the hosts who are interested in watching the same video. After a host selects its interested video, it retrieves a list of hosts currently watching the same video (step 2 in Figure 1). The host then establishes partner relationships (TCP/UDP connections) with a subset of hosts on the list (step 3 in Figure 1). These peers help each other and deliver video traffic cooperatively. The host may also establish a partner relationship with the origin server. Each host viewing the video caches and shares chunks with other hosts viewing the same video. In particular, each host receives *buffer maps (BM)* from its current partners. A buffer map from a remote partner indicates the chunks that are available on that partner. We will discuss the buffer maps in details in Section 2.3. Using a chunk scheduling algorithm, each host requests from its partners the chunks that it will need in the near future. Each host continually seeks new partners from which it can download chunks.

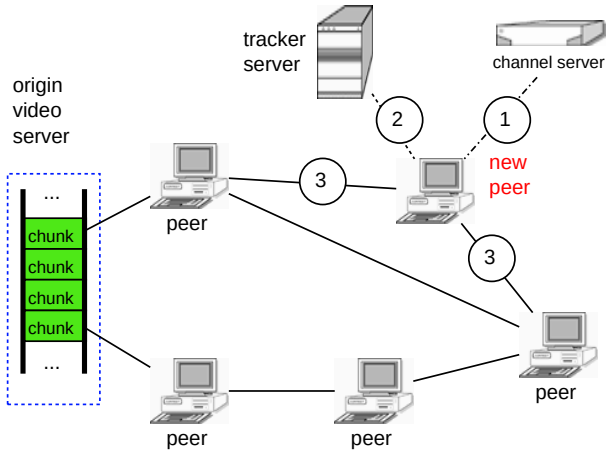


Figure 1: The mesh-pull P2P live streaming architecture

2.2 Software Components

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

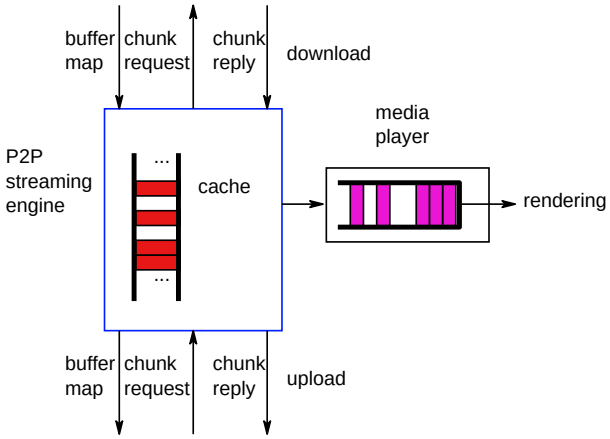


Figure 2: The software components of a peer in a mesh-pull system

2.3 Buffer Map and P2P Streaming Engine

The buffer maps play an important role in our approach to measure the delay performance of mesh-pull streaming systems. At any given time instant, the P2P streaming engine caches up to a few minutes video chunks within a sliding window. Some of these chunks may be chunks that have been recently played; the

remaining chunks are chunks scheduled to be played in the next few minutes. Peers download chunks from each other. To this end, peers send to each other buffer map messages; a buffer map message indicates which chunks a peer currently has buffered and can share. Each chunk is assigned with a unique increasing ID. Specifically, the buffer map message includes the offset (the ID of the first chunk), the width of the buffer map, and a string of zeroes and ones indicating which chunks are available (starting with the chunk designated by the offset), as illustrated in Figure 3. A unique channel ID is also carried in each buffer map message.

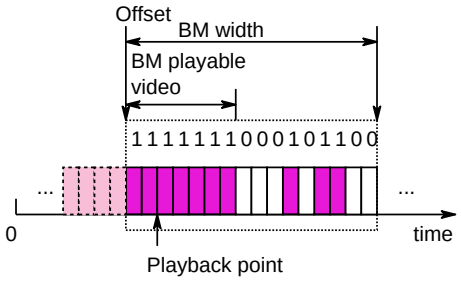


Figure 3: The buffer map structure

A peer can request a buffer map from any of its current partner peers. After peer A receives a buffer map from peer B, peer A can request one or more chunks that peer B has advertised in its buffer maps. A peer establishes partner relationships (TCP/UDP connections) with some peers of the same channel in the network to download video chunks. This peer may also establish a partner relationship with the origin server for chunk download. A peer may download chunks from tens of other peers simultaneously. The streaming engine continually searches for new partners from which it can download chunks. Different mesh-pull systems may differ significantly with their peer selection and chunk scheduling algorithms. The chunks are typically sent over TCP connections, although recently in more mesh-pull systems, video chunks are also transferred using UDP.

2.4 Media Player

When the client application is started, the media player is launched and the URL of the video stream is provided to the media player. As illustrated in Figure 4, from the client's perspective, the server of the media content is the P2P streaming engine (which is in the same host as the media player). Once the media player is activated, it sends (typically) an HTTP request to the P2P streaming engine. After having received the request, the P2P streaming engine assembles the header information and these video chunks in order into a media file, and delivers the file to the media player. Because new chunks continually arrive at the streaming engine, the streaming engine continually adds data to the file. Because some chunks may not arrive before the playback deadline, there may exist "gaps" in the file. When the media player begins to receive the video

from the streaming engine, it buffers the video before playback. When it has buffered a sufficient amount of continuous video content, it begins to render the video.

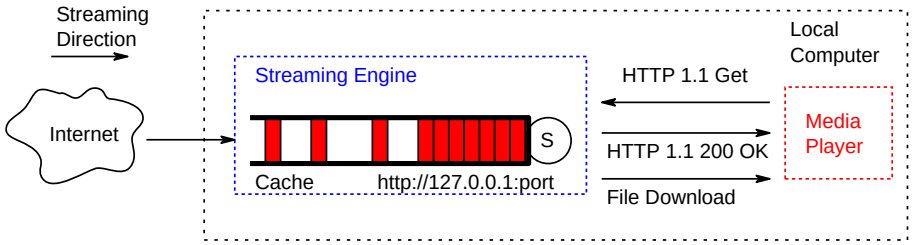


Figure 4: Protocol interaction between the P2P engine and the media player

A video stream is commonly encoded in one of four video formats: Window media video (WMV), Real video (RMVB), Quicktime video or Macromedia flash video. For the sake of concreteness, let us consider mesh-pull P2P streaming systems that use WMV (as is often the case today). WMV videos use the advanced systems format (ASF) [15]. An ASF file starts with a header object and is followed by a series of media data objects. The file header object includes various meta information for the file (i.e., the file identifier, buffering time and overall file bit-rate), number of available audio and video streams carried in the file, and meta information for each audio (i.e., sampling rate) and video stream (i.e., spatial resolution, compression algorithm). Each data object starts with an object header. This object header includes the sequence number of the media object, the number of streams carried in the object, playback duration of the object, and so on. In the payload of data objects, multiple media objects from different streams (for example, a video stream and an audio stream) may be carried and interleaved.

3 Dissecting start-up delay of mesh-pull P2P streaming systems

Three delay metrics are of particular interests in characterizing the delay performance of video streaming systems: start-up delay, channel switching delay and playback time lag among peers. The start-up delay is the time interval from the time instant when one channel is selected until the actual playback starts on the screen. The channel switching delay is the time interval from the time instant when a new channel is selected until the actual playback starts on the screen. The playback time lags among peers indicates the difference of video playback progress on a peer, which is determined by how quickly it collects video chunks from the system. The variations in chunk retrieval times from one peer to another result in time lags among the peers. We are interested in dissecting the start-up delay in this section. Note that in most of the current mesh-pull streaming systems, channels are managed largely independently. Peers form a content delivery swarm and help each other to download video for streaming. When a user switches from one channel to another channel,

this peer essentially repeats the start-up process; hence, the understanding of the start-up delay directly helps to explore effective techniques to decrease the channel switching delay. Nevertheless, in the multi-channel case, common control overlays have been proposed to manage the peers across different channels [16, 17]. These ideas may have applications in the new generation of the P2P streaming systems; then, the channel switching may experience significantly different delay from the start-up. More investigations are required to study the channel switch delay in those cooperative multi-channel overlays.

Before the normal playback, a newly arrived peer ought to actively harvest buffer maps and passively receive buffer maps from neighbors; then, the peer issues chunk requests until the required chunks have been downloaded. Though we have reported the gross start-up delay performance in [10], the understanding of these delay components are not clear. In mesh-pull systems, the fundamental task of the P2P streaming engine is to download video chunks quickly. To this end, a peer should first locate its peer neighbors as potential chunk providers; then, identify which chunks to download. Either its neighbors notify this peer their buffer maps, or this peer acquires the buffer maps of its neighbors, so that the chunk availability information propagates among peers quickly. Finally, this peer "pulls" chunks from its neighbors according to the buffer map information. The amount of the time used in this process often contribute a large portion of the long service delay suffered in mesh-pull systems.

Many factors impact the start-up delay performance of the P2P streaming systems. We are interested in evaluating the impact of peer strategy, chunk scheduling and transport selection on the delay performance of P2P streaming systems. The local peer needs to contact its peer neighbors to download video chunks using an appropriate peering strategy. A good peer selection scheme will help the local peer identify peers with interested contents in a timely fashion. The local peer may download chunks from multiple peers. It remains a question whether the commonly used random chunk scheduling is sufficiently good to provide a low delay performance. Video chunks in mesh-pull streaming systems are carried using TCP or UDP. TCP provides a reliable and congestion aware transport. However, TCP incurs inneglectable connection management overhead. The selection of the TCP transport induces various implications on end-hosts, access links and network cores. Recently we find that mesh-pull streaming systems tend to carry IPTV traffic using UDP instead of TCP. UDP incurs much less connection overhead than TCP; however, this increasing usage of UDP raises the concerns that these UDP traffic should be congestion-aware to avoid network collapse. Because UDP datagrams may be dropped in the networks, streaming applications have to address how to react on packet loss. The simplest strategy is to ignore the dropped UDP datagrams with the expectation that the viewing quality does not suffer too much in spite of a small number of packet dropping; however, too much UDP segment loss will severely degrade the video playback performance. It may be beneficial to retransmit

some lost UDP datagrams with the additional complexity in P2P applications. However, this retransmission may lead to long start-up and playback delay. Our measurement results may be able to reveal insights into the tradeoff between timeliness and reliability.

As shown in Figure 5, we take a closer look at the start-up process. A new peer, interested in viewing the video, requests from the channel server for the available video streams when the P2P software first starts (step 1 in Figure 5). After this peer selects its interested video, it retrieves an initial list of peers currently watching the same video from the tracker server (step 2 in Figure 5). Then this peer starts to contact a sub-set of peers on this initial list to harvest more peers (step 3 in Figure 5). This peer also exchanges buffer maps with its neighbor peers (step 4 in Figure 5). Based on the content of the harvested buffer maps, the peer selects some peers and schedules chunk download from these peer neighbors using some peer selection and chunk scheduling algorithms (step 5 in Figure 5).

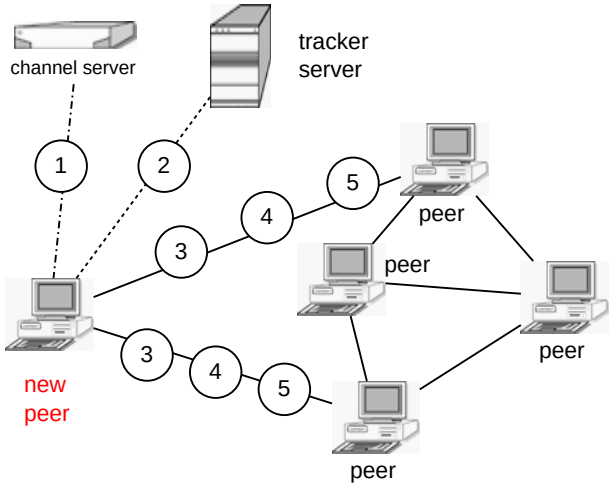


Figure 5: The start-up process of a new peer in mesh-pull P2P live streaming systems

In our measurement experiments, we propose a few delay metrics in characterizing the start-up process. We show these important delay metrics in the time diagram of the signaling exchanges during a normal start-up process in Figure 6. After a user starts the P2P software, it takes t_1 that the user browses the channel list and selects a video channel for viewing. This period of time is largely random depending on user behaviors. During t_1, the channel list may also be updated. In our study, we do not consider this delay t_1. More interesting to us is the delay components after a user selects a channel until the video starts to be played back. As discussed previously, a normal start-up process usually consists of three major steps: harvest peers of the channel, retrieve buffer maps of peer neighbors and download video chunks from neighbors. We denote t_2 be the time difference between the time when the user selects a channel and the time when the local peer receives the first peer list. Note that the first peer list may come from the tracker server or from a normal peer. Then we denote t_3 be the time difference between the time when the local peer receives the first peer list and the

time when the local peer receives the first buffer map. After the first buffer map is received, it takes t_4 to download the first video chunk. After this download time of the first chunk, it takes t_5 to download additional video chunks for caching before the video playback really starts. The streaming engine then activate a local video player when the size of the cached playable video size exceeds a specified pre-buffering video threshold. This threshold is a system parameter of the streaming engine, which an end user may configure in the option settings of the engine. We use the default pre-buffering threshold value specified in the video file header when collecting statistics in our experiments. The streaming engine continues to stream the video chunks to this local player, and the player finally starts the playback after this player finishes the required pre-buffering phase in its local player cache.

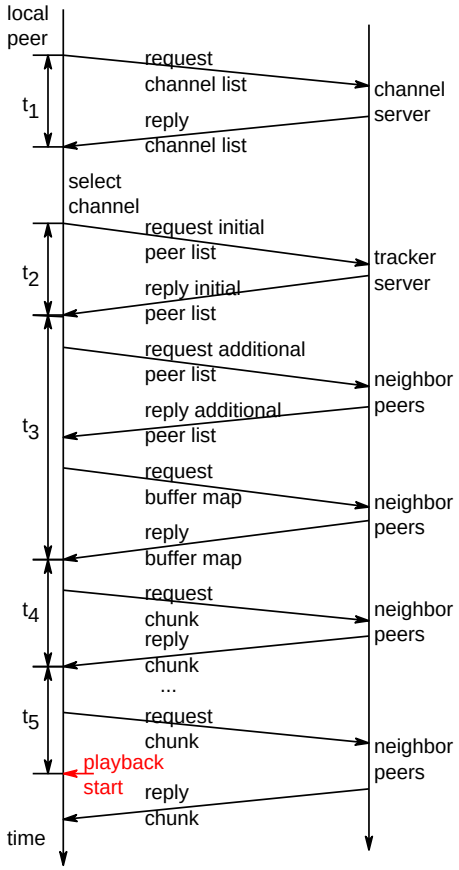


Figure 6: The time diagram of the start-up process of a newly arrived peer in mesh-pull P2P streaming systems

It is relatively easy to determine the values of t_2, t_3 and t_4 by analyzing the traffic pattern based on the protocol knowledge of the corresponding mesh-streaming system. However, it is challenging to correctly determine t_5 since it is unknown for the exact criteria how the P2P streaming engine makes the streaming decision to the local video player. Nevertheless, we can apply various heuristic methods to infer the t_5 value. First, we count the download video chunks in that we are able to trace the exact download time for n-th chunks as time c_n starting from the time instant when the first chunk is downloaded. At the beginning of the chunk download,

chunks may not be able to be downloaded consecutively. The value of c_n does not consider the playback order and the sequence of the downloaded video chunks. Therefore, given a pre-specified buffering threshold θ in terms of video chunk numbers, c_θ may serve as a lower bound of the real video pre-buffering delay t_5. In our experiments, a commonly used pre-buffering video size is 5 seconds in time; therefore, if one chunk size is of 0.1 seconds, θ is roughly 50 chunks.

As demonstrated in [9, 10], buffer maps can also be used to infer the start-up delay when P2P the streaming engine starts up the video player for playback (see Figure 3). When a peer joins the network and has no chunks in its engine buffer, it sends buffer maps with the playback offset of zero and a buffer size of zero. After a peer obtains chunks from other peers, the offset is set to a non-zero value. The consecutive buffer maps provide snapshots of the video chunks in the buffer. Therefore, we can infer the start-up time of the playback of a peer given the sequence of the buffer maps of this peer with the first buffer map having a zero-valued offset. The length of the BM playable video can provide a good indicator when the video playback starts. We denote b_n as the time instant when the BM playable video length is larger than n chunks for the first time after the first chunk is downloaded. Similarly, given a pre-specified buffering threshold θ, b_θ may serve as a good estimate of the real start up delay. In our experiments, to simplify the analysis, we do not re-construct the buffer maps of the local peer based on the downloaded video chunks. Instead, we trace the buffer maps which are sent out by the local peer in its out-going traffic. A side effect is that we may miss the exact b_θ time if there are no buffer maps sent out by the local peer at b_θ. Hence, we approximate b_θ with the sending time of the first buffer map when $K > theta$. Usually, the difference between b_θ and b_K is within sub-second because buffer maps are actively exchanged between the local peer and its peer neighbors.

4 Measurement Methodology

In previous sections, we have provided a background introduction on the system architecture and the signaling protocol of mesh-pull P2P streaming systems. In this section, we outline the setup of our measurement utilities and validate our dissecting methods on the start-up delay.

4.1 Measurement Setup

Our P2P network measurements on PPLive fall into two categories: passive sniffing and active crawling. The passive sniffing is used to gain a deeper insight into PPLive from the perspective of residential users and campus users. The active crawling is used to obtain user behaviors and the global view of the entire PPLive network for any channel. As shown in Figure 7, we captured the incoming and outgoing traffic from the local

peer using the traffic capture utility Wireshark [18]. We also correlate the video start-up performance with various network factors.

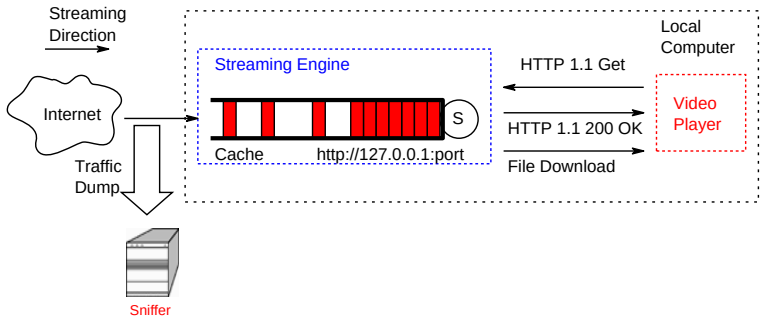


Figure 7: PPLive traffic sniffing

We sniffed and collected multiple PPLive packet traces from four PCs shown in Table 1: one PC connected to the campus network of Polytechnic Institute of NYU with the 100 Mbps Ethernet access; one PC connected to the residential network through cable modem located at Brooklyn, New York; and two PCs connected to the campus network of Hong Kong University of Science and Technology with the 100 Mbps Ethernet access. Most of the PPLive users today have either one of these two types of network connections. Each PC ran Wireshark to capture all the 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. To have an overview about the peer dynamics in the monitored channels, we also launched PPLive peer list crawlers in two locations: one PC connected to the campus network of Polytechnic Institute of NYU with the 100 Mbps Ethernet access and one PC connected to the campus network of Hong Kong University of Science and Technology with the 100 Mbps Ethernet access. This peer list crawler implements the peer tracking methodology by exploiting PPLive's distributed gossiping algorithm [9]. The peer crawling is conducted in a round-by-round fashion consecutively. In each round, the aggregate peer list harvested by the crawler includes all the peers of the channel. At the beginning of the next crawling round, the crawling is restarted from a fresh state. By investigating the peers on the aggregate peer list harvested in each crawling round, we are able to determine the peer arrivals and departures. This may lead to some inaccuracy of determining the exact peer arrival and departure time. However, this inaccuracy is constrained at the time granularity within the minute level.

To study the start-up delay, we have collected traces for a number of live channels with different popularity as shown in Table 2. The "x-star" popularity in the table is marked for each PPLive channel. This popularity parameter is computed based on the peer statistics and is provided by the PPLive web site. We use it for a reference of the channel popularity. It is not feasible to conduct extensive measurements on start-up delays by

Table 1: Summary of sniffers and crawlers

	IP	Access	location
Sniffer	128.238.88.50	campus	New York
	24.185.66.10/192.168.123.110	cable	New York
	143.89.47.138	campus	Hong Kong
	143.89.47.217	campus	Hong Kong
Crawler	128.238.88.52	campus	New York
	143.89.47.160	campus	Hong Kong

initiating multiple start-up processes manually. We instrument scripts to automate the PPLive experiments to initiate multiple start-up sessions for the interested channels. The scripts initiate the PPLive client to connect to one selected channel, keep it downloading video and connected to this channel for 5 minutes; then, shut down the clients and remain silent for 20 minutes. This start-up on-off process is repeated for 20+ times for each channel. We captured the incoming and outgoing traffic of the monitored peer. Therefore, for each channel, 20+ start-up sessions are captured with traces. Based on our understanding of the PPLive protocol, we successfully dissect different delay components of the start-up process of newly arrived PPLive peers in a channel by tracing the PPLive signaling and video traffic. This traffic analysis of the captured traces shed some light on the major portions of start-up delays.

Table 2: Summary of the monitored channels

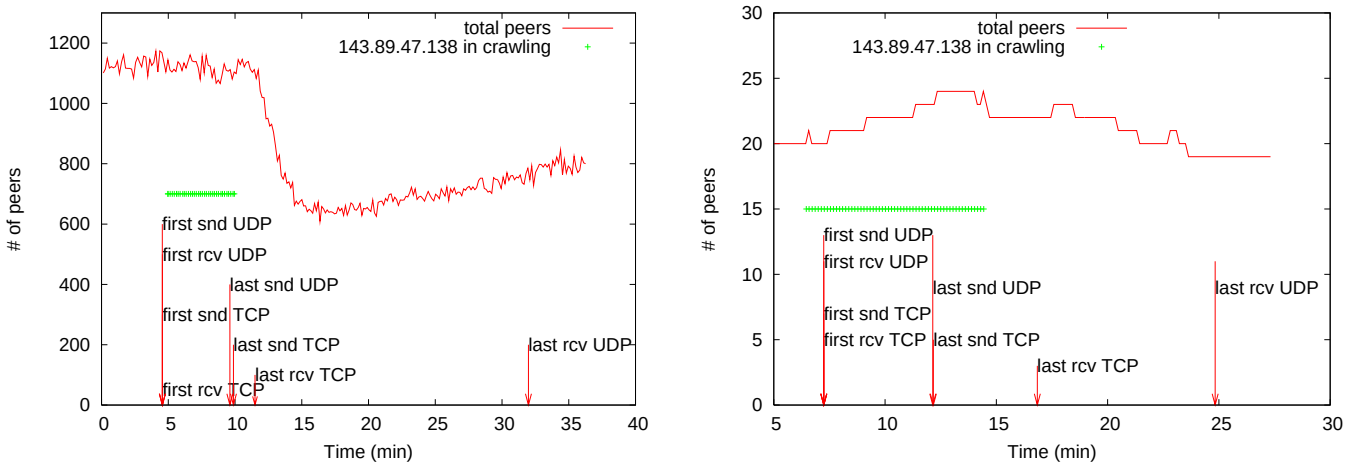
channel name	type	popularity	bit rate (kbps)	chunk size (byte)
Hunan TV	TV Live	4-star	381	4818
DragonTV	TV Live	2-star	390	4793
Hubei TV	TV Live	1-star	381	4818
TianjinTV	TV Live	1-star	381	4793
DragonBall	Cartoon	5-star	381	7239
Conan	Cartoon	3-star	381	7239
VNETGX	Cartoon	0-star	434	5369
PGLS3-SC	Game show	4-star	400	4918
ROTK08-WAR3	Game show	4-star	400	4918
LuDingJi	TV play series	4-star	381	7239
RedMansions	TV play series	1-star	380	4669

4.2 Validation

We captured two types of the start-up traces: single start-up and multiple start-ups. For single start-up experiments, we initiate one complete PPLive start up session manually in each experiment and capture the traffic trace of the local peer. By studying the detailed signaling and video traffic messages between the local peer and the other peer neighbors, we apply the delay dissection criteria and determine various delay

measurement values, t_2, t_3, t_4, b_n and c_n.

As shown in Figure 8, we plot the start-up process that a peer with the IP address 143.89.47.138 joins two channels respectively. The channel ROTK08-WAR3 is a popular game channel with the channel size around 1000 peers during the measurement period. The channel VNETGX is a unpopular cartoon channel with the channel size around 20 peers. The green points in the figure indicate the time instants when this peer appears first in the harvest peer list in each crawling round. Note that these green points are plotted for the illustration purpose, in which the y-coordinates of these points are of no physical meaning and only their x-coordinates show the meaningful time instants. The PPLive client was shutdown around 5 minutes after the start-up. We can observe that when the client is connected to the PPLive network, this client remains in the peer list crawling traces. After the client is terminated, this client no longer appears in the crawled IPs. The signaling and video transport are carried using TCP and UDP in PPLive. As show in Figure 8, after the PPLive client leaves the channel, the local peer still keeps receiving the UDP signaling messages for up to around 20 minutes. These UDP messages usually carry the peer list information.



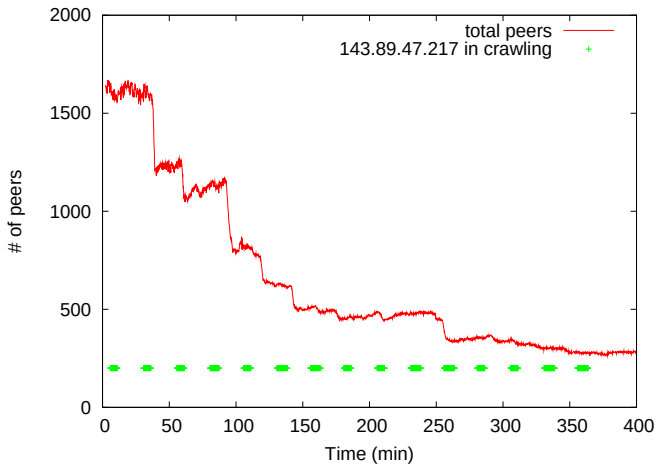
(a) channel: ROTK08-WAR3

(b) channel: VNETGX

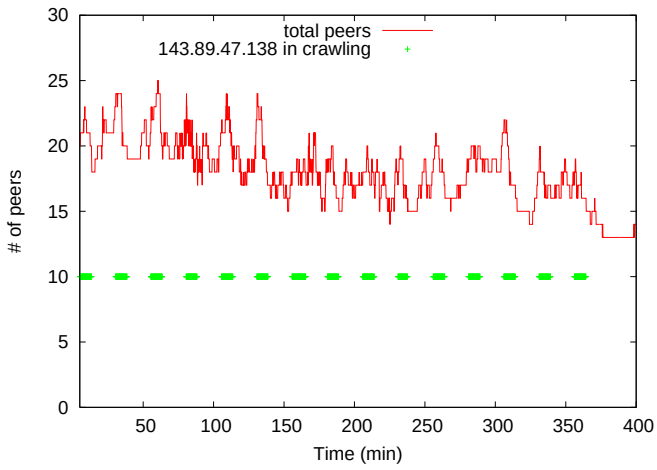
Figure 8: Single start-up sessions

In Section 4.1, we set the off time period between two start-up sessions to be 20 minutes. Based on the trace analysis of the single start-up sessions, this setting is configured to ensure that two consecutive start-up sessions are not interfered with each other with signaling and video traffic. Similarly, the green points in the figure indicate the time instants when this peer appears first in the harvest peer list in each crawling round, as shown in Figure 9. These green points show a clear on-off client start-up pattern in our experiments.

We also evaluate the accuracy of estimating t_5 using the BM playable video size b_n in buffer maps. Using our monitored peers, we manually recorded two types of start-up delays in PPLive: the delay from when one



(a) channel: ROTK08-WAR3



(b) channel: VNETGX

Figure 9: Multiple start-up sessions

channel is selected until the streaming player pops up; and the delay from when the player pops up until the playback actually starts. As shown in Table 3, b_{50} is usually less than t_5 and hence provides a conservative estimation.

Table 3: Human validation on video playback of PPLive v1.0.9.6

channel name	location	channel selection	player popup	playback start	player popup delay (sec)	player buffering delay (sec)	b_{50} (sec)
Hunan TV	HK-Campus	23:43:23	23:44:17	23:45:19	54	116	70.79
ROTK08-WAR3	HK-Campus	23:12:15	23:13:07	23:14:52	52	157	86.29
LuDingJi	HK-Campus	23:19:25	23:21:08	23:21:46	103	141	56.02
RedMansions	HK-Campus	23:30:50	23:31:31	23:32:32	41	102	87.88

5 Start-up delay measurement results

In this section, we report the start-up delay measurement results on the single start-up sessions. We will further correlate various network factors with the measured start-up delay in Section 6. In addition, we also provide the measured delay statistics of automated multiple start-up sessions to have a more general understanding of the start-up delay performance of PPLive.

5.1 Single start-up sessions

As tabulated in Table 4, the time to harvest the first peer list, t_2, is usually very small (< 1 seconds); the time to retrieve the first buffer map, $t3$, is at the time granularity of the second level; the time to download

the first video chunk, t_4, is also at the time granularity of the second level. We use c_n and b_n to estimate the time duration to download a pre-specified number of video chunks, t_5. As shown in Table 5 and 6, t_5 is at the time granularity of 10+ seconds. On the other hand, c_n and b_n only provide conservative estimates of t_5. A more accurate estimation of t_5 is to re-construct the pre-buffering streaming process of the P2P engine based on the chunk download process extracted from the traffic traces.

Table 4: Signaling delay in single start-up experiments

ID	channel name	location	t_2(sec)	t_3 (sec)	t_4 (sec)
1	Hunan TV	HK-Campus	0.05	0.59	0.54
2	Hunan TV	HK-Campus	0.00	2.97	1.77
5	DragonTV	HK-Campus	0.00	1.95	16.40
6	TianjinTV	HK-Campus	0.00	1.48	2.89
7	DragonBall	HK-Campus	0.00	2.43	2.77
8	DragonBall	HK-Campus	0.18	-0.15	0.18
9	Conan	HK-Campus	0.03	0.81	0.47
10	VNETGX	HK-Campus	0.00	1.47	2.75
11	VNETGX	HK-Campus	0.26	-0.02	0.12
12	PGLS3-SC	HK-Campus	0.00	2.43	2.42
13	ROTK08-WAR3	NY-Cable	0.00	3.02	2.30
14	ROTK08-WAR3	HK-Campus	0.15	-0.07	0.09
15	ROTK08-WAR3	HK-Campus	0.00	3.30	0.84
16	ROTK08-WAR3	HK-Campus	0.00	9.07	2.10
17	ROTK08-WAR3	HK-Campus	0.00	2.95	1.35
18	ROTK08-WAR3	HK-Campus	0.00	3.49	1.72
19	LuDingJi	HK-Campus	0.03	0.08	1.14
20	LuDingJi	HK-Campus	0.00	3.65	0.66
21	RedMansions	HK-Campus	0.08	0.11	0.49

5.2 Multiple start-up sessions

In this sub-section, we further provide the comprehensive start-up delay statistics for a number of the monitored channels with 20+ start-up sessions for each channel. In Table 7, we tabulate the average and the standard deviation of the measurement delay components in the start-up process. The measurement results in this table confirm again that major delay bottleneck is on the chunk download time for pre-buffering a sufficient number of video chunks.

We also plot the cumulative density function (CDF) of these delay components in Figures 10, 11 and 12. We can observe that the worst chunk download time for pre-buffering may reach over 100+ seconds. In Figure 10, the signaling delays, including harvesting peer list and downloading buffer maps, only contribute a small portion of the total start-up delay. Nevertheless, these delays are still at the time granularity of seconds. If P2P video streaming services are eventually able to compete with traditional television services, the start-up

Table 5: Video pre-buffing time estimation for t_5 by counting the downloaded chunks in single start-up experiments

ID	channel name	location	c_{10}(sec)	c_{50} (sec)	c_{100} (sec)
1	HunanTV	HK-Campus	2.62	9.42	13.43
2	HunanTV	HK-Campus	1.32	5.81	10.99
3	HunanTV	HK-Campus	0.51	2.63	5.84
4	DragonTV	HK-Campus	2.62	9.47	16.09
6	TianjinTV	HK-Campus	1.43	11.36	28.73
7	DragonBall	HK-Campus	2.81	8.27	13.37
8	DragonBall	HK-Campus	0.56	4.02	7.03
9	Conan	HK-Campus	3.05	7.30	13.95
10	VNETGX	HK-Campus	1.67	5.54	10.96
11	VNETGX	HK-Campus	0.79	5.49	11.05
12	PGLS3-SC	HK-Campus	0.89	5.81	10.80
13	ROTK08-WAR3	NY-Cable	0.87	3.49	5.08
14	ROTK08-WAR3	HK-Campus	2.75	8.96	15.30
15	ROTK08-WAR3	HK-Campus	1.06	5.69	10.44
16	ROTK08-WAR3	HK-Campus	5.39	20.12	43.23
17	ROTK08-WAR3	HK-Campus	1.51	9.66	15.31
18	ROTK08-WAR3	HK-Campus	5.76	11.04	15.40
19	LuDingJi	HK-Campus	2.83	10.66	19.22
20	LuDingJi	HK-Campus	3.31	9.11	19.43
21	RedMansions	HK-Campus	2.11	9.25	18.13
22	RedMansions	HK-Campus	0.58	4.60	9.22

delay should strive to be less than a few seconds in order to achieve similar video surfing experiences.

Based on these measurement results, the chunk pre-buffering time appears to the major delay bottleneck in the PPLive start-up process. Nevertheless, it remains unclear whether the peers in the initial harvest peer lists are "good peers", in that they are able to provide useful video chunks for the local peer to download. It may take a long time for those "good peers". In the next section, we will analyze the captured traces to evaluate the quality of the peer neighbors and provide a quantitative illustration on the search delay for these peers.

6 Understanding start-up delay of PPLive

In this section, we analyze the signaling traffic and the download process of PPLive video chunks of a campus peer in a popular channel PGLS3-SC in order to understand the reasons of the long start-up delays of PPLive clients. First, we will evaluate the signaling efficiency of harvesting peer lists of PPLive clients using some gossip protocol, then assess the promptness of the video chunk buffering process of PPLive peers and finally review the video chunk download performance of peer neighbors. To assist the discussions, we set the time

Table 6: Video pre-buffing time estimation for t_5 using buffer maps in single start-up experiments

ID	channel name	location	b_{10}(sec)	b_{50} (sec)	b_{100} (sec)
1	HunanTV	HK-Campus	77.82	77.82	77.82
2	HunanTV	HK-Campus	8.29	70.29	178.26
3	HunanTV	HK-Campus	3.79	58.74	58.74
4	DragonTV	HK-Campus	24.76	31.77	65.17
5	DragonTV	HK-Campus	27.66	27.66	27.66
6	TianjinTV	HK-Campus	3.70	73.71	101.68
9	Conan	HK-Campus	20.70	122.42	122.42
10	VNETGX	HK-Campus	5.81	21.80	116.84
11	VNETGX	HK-Campus	7.68	64.58	64.58
12	PGLS3-SC	HK-Campus	13.17	58.23	184.15
13	ROTK08-WAR3	NY-Cable	5.69	73.67	87.67
14	ROTK08-WAR3	HK-Campus	71.82	191.82	191.82
15	ROTK08-WAR3	HK-Campus	4.88	98.90	117.86
16	ROTK08-WAR3	HK-Campus	18.91	74.88	74.88
18	ROTK08-WAR3	HK-Campus	72.80	85.79	85.79
19	LuDingJi	HK-Campus	72.79	88.50	88.50
20	LuDingJi	HK-Campus	55.96	55.96	55.96
22	RedMansions	HK-Campus	10.83	87.80	94.12

zero as the sending time of the channel registration message of the local peer in the figures in this section.

In Section 5, we have conjectured that the long video buffering delay is mainly due to the difficulty for the local PPLive client to locate "good" video peers, from which the local peer is able to download video chunks with high rates. We start with an evaluation on the peer quality of the peer list harvested by the local peer. The peer quality is used to quantify whether a peer neighbor is able to provide a high video download rate for the local peer. As we shown in Section 2, after a user starts the PPLive client software, the local peer first harvests an initial peer list from the tracker server and contacts the peers on this initial list for more peer lists using some gossip protocol recursively; then, it retrieves buffer maps from these peer neighbors; finally, it

Table 7: Start-up delay statistics of multiple start-up sessions across channels

channel name	t_2 avg	t_2 std	t_3 avg	t_3 std	t_4 avg	t_4 std	b_{50} avg	b_{50} std
HunanTV	0.023	0.041	2.49	2.15	0.60	1.91	68.4	42.7
Hubei TV	0.061	0.097	1.94	1.76	0.90	0.96	64.1	53.3
DragonTV	0.094	0.15	1.66	1.50	1.01	1.04	58.3	44.7
TianjinTV	0.21	0.44	1.58	1.73	0.76	0.86	58.1	49.2
DragonBall	0.33	0.099	2.32	1.43	1.40	1.22	42.7	52.3
Conan	0.029	0.053	1.74	1.55	1.05	1.11	55.7	42.6
VNETGX	0.062	0.11	1.38	1.26	1.07	1.11	28.3	39.2
ROTK08-WAR3	0.065	0.27	1.91	1.60	0.97	1.04	83.5	40.1
LuDingJi	0.056	0.12	2.03	1.51	1.04	0.97	69.4	69.0

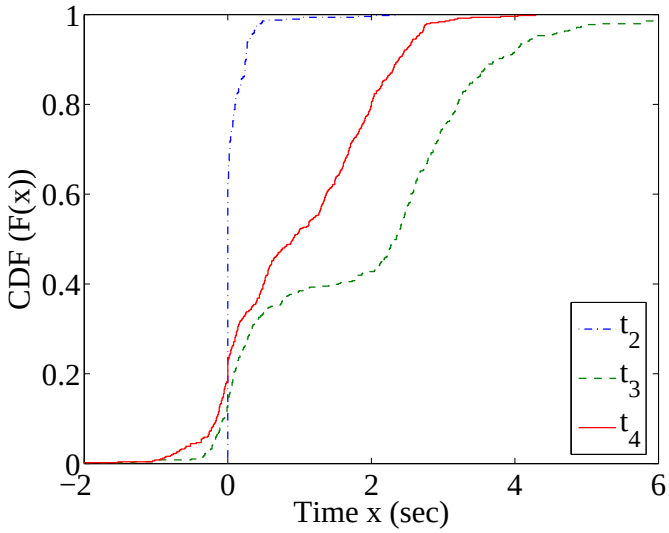


Figure 10: Delay component t_2, t_3 and t_4 across all monitored channels

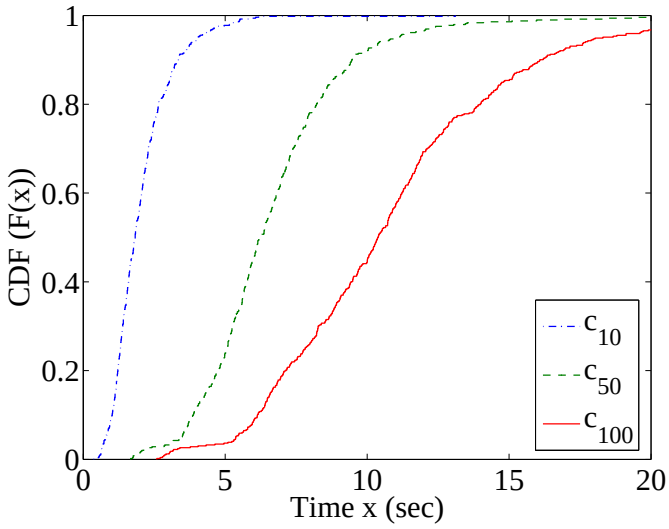


Figure 11: Delay component c_{10}, c_{50} and c_{100} across all monitored channels

starts download video chunks from some selected neighbors. As shown in Figure 13, we evaluate the signaling efficiency of harvesting peers list in channel PGLS3-SC. The "all" curve is the aggregate harvested peer list from the tracker server and the peer neighbors of the controlled peer in channel PGLS3-SC. Figure 13 shows that a PPLive peer keeps harvesting peer lists from the PPLive network and the peer number of this aggregate peer list increases quickly as the time increasing. Within 300 seconds, the total number of the harvested peers may reach 1000+ for popular channels such as PGLS3-SC. These peers are the potential video chunk providers for the local peer. PPLive clients utilize both UDP and TCP to exchange peer information. The "UDP send" curve indicates the number of peers, to which the local peer sends UDP signaling packets. The "UDP receive" curve shows the number of peers, from which the local peer receives UDP signaling packets. We can observe a large gap between the "UDP send" curve and the "UDP receive" curve, in particularly at the beginning

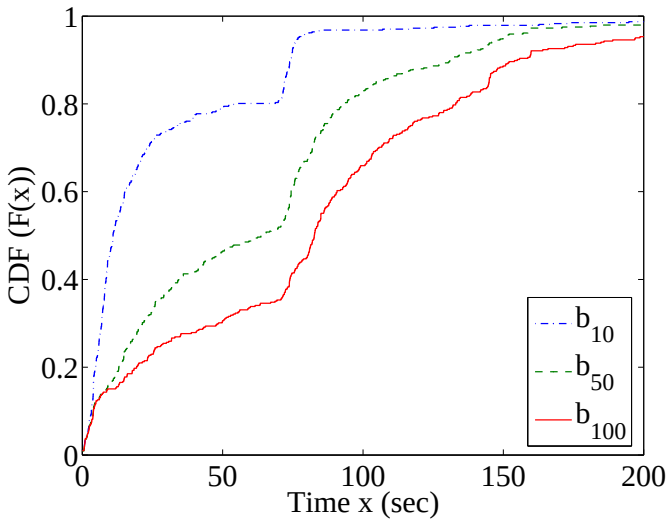


Figure 12: Delay component b_{10}, b_{50} and b_{100} across all monitored channels

of the start up process. This gap implies that the peers on the harvested peer list may not be very updated in that many peers may have already departed from the network; therefore, the local peer contacts a large number of peers using UDP, only a small percentage of the peers respond. Another possible reason of this small peer response ratio is due to the network address translation (NAT) traversal issue in that UDP packets are blocked by NAT servers or firewall servers. Nevertheless, as the time increases at 300 seconds, the gap between the "UDP send" curve and the "UDP receive" curve becomes quite small. This is a good indication that NAT traversal may not be a significant problem for the PPLive network. Instead, it just takes a long time for the local peer to discover a sufficiently large number of peers for video download. PPLive clients also utilize TCP to exchange peer lists and accomplish other signaling functions. The "TCP send" curve plots the number of peers, to which the local peer sends TCP signaling segments. The "TCP receive" curve shows the number of peers, from which the local peer receives TCP signaling segments. Unlike UDP, the gap between the "TCP send" curve and the "TCP receive" curve is very small due to the fact that TCP is a connection-oriented protocol, in which either end-point can initiates the connection and utilize the same connection for exchanging signaling segments. Nevertheless, the existence of the gap between the "TCP send" curve and the "TCP receive" curve indicates that some peers are solely the receivers of TCP signaling segments.

We further classify the peers on the harvested peer list into different types. Figure 14 shows the process for the local peer to locate video peers in channel PGLS3-SC. The "server" curve shows the number of peers on the aggregate harvested peer list retrieved from the tracker server. The "neighbor" curve shows the number of peers on the aggregate harvested peer list retrieved from peer neighbors. We can see that at the beginning of the start-up process, the local peers harvested 100+ peers from the tracker server and the major harvested peers are retrieved from the peer neighbors using some gossip protocol. The "buffermap" curve shows the

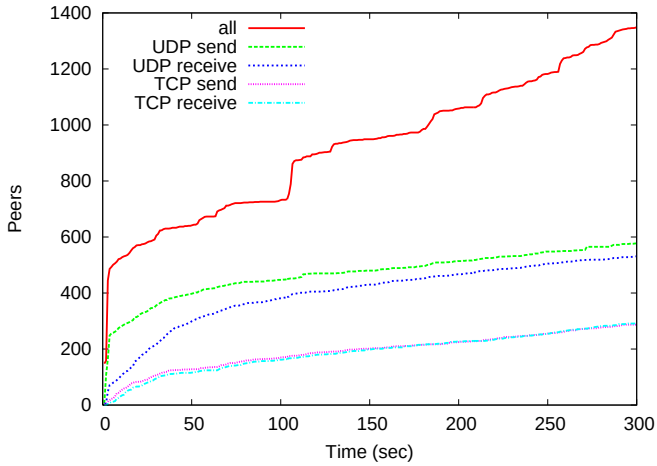


Figure 13: Signaling efficiency of harvesting peers list in channel PGLS3-SC

number of peers, which send buffer maps to the local peer. The "chunk" curve shows the number of peers, which send video chunks to the local peer. It appears that PPLive clients contact dozens of peers and harvest a sufficient number of buffer maps before it starts to download video chunks. These buffer maps provide a clear picture about the video chunk availability in the caches of the peer neighbors; hence, the local peer may crank out a good starting point to download video chunks. Nevertheless, in Figure 14, the local peer appears not to be able to locate good video peers quickly. At $t = 50$ seconds, the local peer only downloads video chunks from only 3 peers. At $t = 150$ seconds, the video peers only increases to 4.

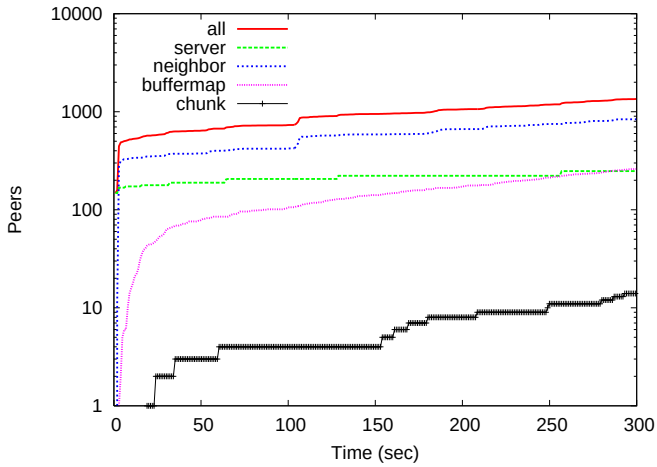


Figure 14: Locate video peers in channel PGLS3-SC

The number of video peers increase slowly in the start-up process. Their contribution of video chunk download also differs significantly. We use the following criterion to characterize the peer quality in terms of its video chunk download performance. Given m video peers and n video chunks, the average chunk download per peer is n/m. Denote the number of download chunk from peer i as d_i. If $d_i > n/m$, we classify peer i as a "good" video peer. Figure 15 demonstrates the difficulty that the local peer finds good video peers. In

Figure 15, the "active peer" curve shows the number of peers, to/from which the local exchange signaling packets or video traffic. The number of active peers increases to 100+ within just a few seconds; however, the local peer can only contact a very small number of peers for video download. An even smaller number of peers download more chunks than the average chunk download per peer so as to stand out as a "good" video peer. In particular, at around $t = 175$ seconds, $t = 200$ seconds and $t = 248$ seconds, only 1 peer contributes significantly larger downloads than other peers.

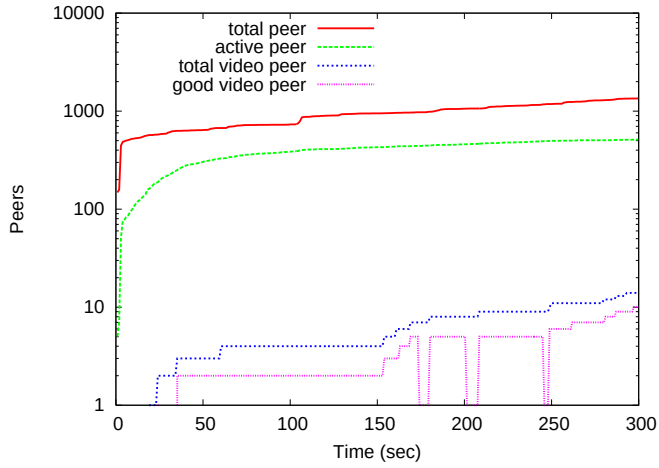


Figure 15: Chunk download from peers in channel PGLS3-SC

PPLive peers download video chunks from multiple peers in order to sustain a sufficient high download rate for video playback. However, the contribution of the peers may differ significantly. Figure 16 shows the peer download performance of a campus peer in channel PGLS3-SC. The "top n" curve is the aggregate chunk download ratio of the total download for the peer with the largest n chunk download volume up to time t. For this campus peer in Figure 16, one peer neighbor contributes the total chunk download for the first 20 seconds after the peer starts up. Up to 150 seconds, the top 3 peers contribute over 80% of the total chunk download.

Figure 17 shows the video download rate of a campus peer in channel PGLS3-SC. The instant rate is computed with the time average interval of 1 second. The cumulative rate is computed as the ratio between the total chunk download and the total download time t. We observe that after $t = 50$ seconds, the local peer is able to achieve quite a stable high download rate. It is a good indication that the local peer can achieve a smooth video playback. This is also consistent with $b_{50} = 58.23$ seconds of channel PGLS3-SC in Table 6. In Table 2, for channel PGLS3-SC, the media playback bit rate is 400 kbps and the chunk size is 4918 bytes; the required chunk download rate for smooth video playback should be at $r = 400 \cdot 1000/(4918 \cdot 8)$ chunks/sec $= 10.2$ chunks/sec. The instant rate and the cumulative rate of the video download tends to be larger than this nominal playback rate. This is counter intuitive; nevertheless, a closer examination of the downloaded video

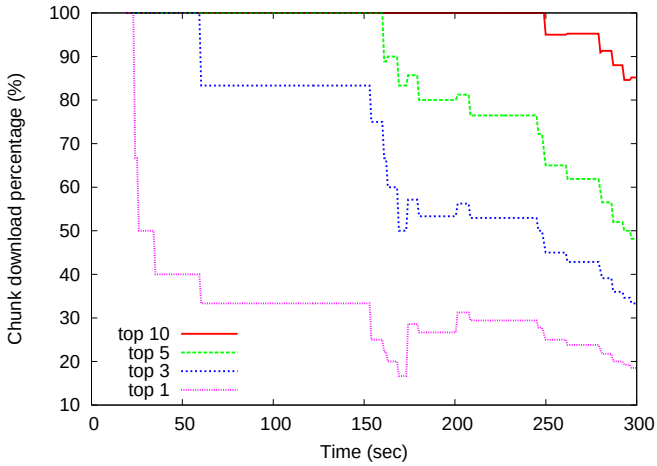


Figure 16: Peer download performance in channel PGLS3-SC

chunks leads to the finding that PPLive clients may download duplicate video chunks during the start-up process.

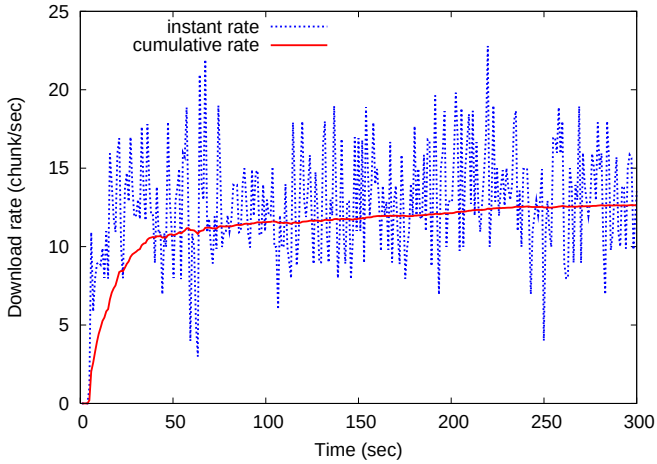


Figure 17: Video download rate in channel PGLS3-SC

We apply the video start-up heuristic in that the video playback starts at b_{50}. Table 8 shows the download chunk statistics at the time of b_{50} for each channel. As discussed in Section 2, each chunk is associated with a unique increasing chunk ID. We trace the IDs of the downloaded chunks and find that there exist a large number of duplicate download chunks. In Table 8, at the time of b_{50} for channel PGLS3-SC, we recorded a total 684 downloaded chunks, in which 104 chunks have duplicate chunk IDs with one or more chunks. Hence, the chunks with unique IDs is only 580. We also observe that the BM video size is 88, which can be played back continuously. However, the total buffered video chunks already reach 579. It appears that PPLive clients apply conservative buffering strategies in that they download more chunks than the required pre-buffering video size. Considering the total buffer size is 1488, the number of chunk holes in the buffers may still be quite large. In Table 8, we also count the number of video chunks with the ID value smaller than the offset of

the buffer map at the time b_{50}. Because these chunks are not included in the buffer maps, we consider them as dropped chunks by the local peer before the playback. Note that in our traced chunks, we find that some chunks have chunk IDs, which are largely away from the ID range of the other chunks. We consider these chunks as errors chunks and exclude them in our analysis. We conjecture that there may be two reasons for the existence of the error chunk IDs. One is due to the possible software bugs in the PPLive client and the chunk IDs is labeled incorrectly when these chunks are sent out. The other may be due to a false video chunk tracing of our PPLive traffic analyzer. Nevertheless, the number of error chunks remains small or even zero across channels. Our analysis results may not deviate too much due to these error chunks.

Table 8: Download chunk statistics at the time of b_{50}

ID	channel name	total chunk	effective chunk	duplicate chunk	dropped chunk	error chunk	buffered chunk	buffer size	BM video size
1	HunanTV	1670	1422	248	69	4	1349	1440	113
2	HunanTV	796	683	113	1	4	678	1440	51
4	DragonTV	259	250	9	1	7	242	904	51
6	TianjinTV	252	240	12	60	1	179	1144	64
7	DragonBall	1178	1005	173	1	0	1004	1864	56
9	Conan	1190	855	335	1	7	847	1120	112
10	VNETGX	199	176	23	1	0	175	984	62
11	VNETGX	579	549	30	1	0	548	1024	128
12	PGLS3-SC	684	580	104	1	0	579	1488	88
13	ROTK08-WAR3	1647	1418	229	39	0	1379	1424	72
14	ROTK08-WAR3	2689	2112	577	702	3	1407	1448	197
15	ROTK08-WAR3	1420	1094	326	156	0	938	976	85
17	ROTK08-WAR3	1968	1640	328	507	1	1132	1584	160
18	ROTK08-WAR3	1004	819	185	154	2	663	1304	384
19	LuDingJi	480	418	62	61	8	349	624	120
20	LuDingJi	477	388	89	0	0	388	840	120
22	RedMansions	1195	910	285	158	3	749	1272	96

7 Reducing start-up delay of P2P streaming systems

In this section, we provide an overview on various possible approaches to reduce the start-up delay of P2P streaming systems. The measurement results in Section 5 and the analysis in Section 6 show long start-up delays and the delay bottleneck is on finding "good peers" for streaming video content for the local peer. These peers may not be in the initial peer list. The baseline solution is to download chunks from the servers after each start-up and each channel switch, until the local peer finds satisfactory peers, and then offload from the servers. To be cost-effective, it is crucial that service providers detect quickly when the service quality degrades, so that they can add, retroactively and dynamically, additional uploading capacity. When

the service quality is satisfactory, the infrastructure capacity can be released for other purposes. A more cost-effective approach is to download initial chunks from stable super peers, which stick to the channel. The percentage of these stable super peers in the P2P streaming networks may vary significantly; however, they have been playing important role in contribute significantly to video data uploading [9, 19]. In the current mesh-pull streaming systems, these super-peers usually evolve passively. If new P2P streaming systems initiate proactive actions in identifying these nodes and move more network functions onto these nodes, the system performance may be improved significantly including reducing the start-up delay.

As discussed in Section 2, the current P2P streaming systems usually support many video channels. Those peers, who are interested in the same video content, join the same channel for cooperative video streaming. The peers are managed and cooperated on a channel-basis. Peers of different channels will not help each other in signaling and video traffic transport. Therefore, the breakdown of one channel will not interfere with another channel. Nevertheless, in this multi-channel case, common control overlays may be very effective to manage the peers across the channels in providing high-quality peer and video chunk information; hence, the start-up delay can be reduced considerably.

Another avenue in reducing start-up delay is on the system architecture. Unlike mesh-pull systems, tree-push systems have smaller delay performance when the tree structure does not break down due to peer churn and peers at the higher level of the tree have sufficient upload capacity to support the streaming of their children peers. The debate on whether mesh-pull or tree-push is more suitable for IPTV is still on-going [20] [21]. The architecture design tradeoffs will impact various system characteristics significantly [22], i.e., system resilience and signaling overhead, etc.. Video can also be divided into chunks and be delivered via tree-push systems. Before the chunk delivery starts, the single/multiple-tree structure is established and the chunks are just forwarded from the root of the tree at the video source, to the different level of tree hierarchy. There are no further communication overhead for the subsequent chunk delivery unless the trees have to be repaired due to the departure of intermediate peers. Therefore, the delay performance in tree-push systems is in general good when the trees are stable.

From the technology side, we may strive research efforts to minimize start-up delays for high-quality video streaming services, another interesting direction is to optimize the streaming application to achieve satisfied user experiences. In addressing noticeable start-up delay, some advertisement videos can be pre-cached on the users' side; when a user starts the streaming application, immediately these advertisement videos are played back. During the playback of these commercials, videos are downloaded at the background. Therefore, users experience zero start-up delay, while the video streaming service provider finds an avenue to obtain revenues. Another possibility is to provide multiple videos with different bit rates for the same channel. If dedicated

servers are deployed, they can be used to provide the low-quality video, and the P2P delivery architecture is utilized to transfer the high-quality video. When a user joins the service, the P2P engine downloads the low-quality video from the server to achieve a quick service initiation; then, the engine downloads the high-quality video from other peers. Technologies may provide the foundation of the video streaming application; nevertheless, a deep psychological understanding on user behaviors and expectations are helpful in optimizing the application to maximize the user viewing experience.

8 Conclusion and future work

In this paper, we conducted an in-depth measurement study on characterizing the start-up delay performance of a popular mesh-pull streaming system, PPLive. Our measurement results show that 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 services. We dissect the start-up delay and correlate various factors with the measured start-up delay components in order to reveal the major delay bottleneck. We find that the major reason of the long start-up delay is due to the slow speed in finding "good" video peers for the local peer to download video contents.

As discussed in Section 7, infrastructured servers and super-peers may be very helpful in reducing start-up delay. The core research issue is to characterize how much load is imposed on the servers, given an aggregate start-up delay (across all users). We are interested in evaluating the feasibility of this approach. In particular, based on the peer crawling traces, we will examine the existence of stable super peers; then, examine the usage of those super peers in the system using trace-driven simulations.

We also intend to explore the possible new mesh-pull schemes to release the delay bottleneck and eventually to improve the delay performance. This mesh-pull scheme enjoys the signaling simplicity and performs as well as in terms of delay as those proposed push-pull schemes. We will compare the proposed mesh-pull scheme with some of the existing push-pull schemes. The comparison study is conducted via trace-driven simulations. A trace-driven simulation study may provide a common evaluation platform to examine the architecture design tradeoff on the delay performance. We developed a trace-driven simulator based on the P2P streaming simulator reported in [21] with the PPLive traces serving as the input of the simulator, in order to capture the impact of real P2P workload on the start-up delay performance.

Acknowledgment

This work is supported by the Nanjing research institute of Huawei Technologies Co., Ltd..

References

[1] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end-system multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

[2] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, vol. 3, Miami, FL, USA, Mar. 2005, pp. 2102 – 2111.

[3] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 11–24, Jan. 2008.

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

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

[6] "UUSee," http://www.uusee.com.

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

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

[9] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.

[10] X. Hei, Y. Liu, and K. W. Ross, "Inferring network-wide quality in P2P live streaming systems," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 10, pp. 1640–1654, Dec. 2007.

[11] A. Reibman, S. Sen, and J. V. der Merwe, "Video quality estimation for Internet streaming," in *Proc. ACM WWW: Special interest tracks and posters of the 14th international conference on World Wide Web*, Chiba, Japan, 2005, pp. 1168–1169.

[12] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang, "Delving into Internet streaming media delivery: a quality and resource utilization perspective," in *Proc. ACM SIGCOMM conference on Internet measurement (IMC)*, Oct. 2006, pp. 217–230.

[13] S. Agarwal, J. P. Singh, A. Mavlankar, P. Baccichet, and B. Girod, "Performance and quality-of-service analysis of a live P2P video multicast session on the Internet," in *Proc. 16th IEEE International Workshop on Quality of Service (IWQoS)*, June 2008, pp. 11–19.

[14] S. Tao, J. Apostolopoulos, and R. Guerin, "Real-time monitoring of video quality in IP networks," *IEEE/ACM Transactions on Networking*, Dec. 2008.

[15] "Advanced systems format (ASF) specification," 2004. [Online]. Available: http://www.microsoft.com/windows/windowsmedia/forpros/format/asfspec.aspx

[16] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of BitTorrent-like peer-to-peer systems," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 1, pp. 155–169, Jan. 2007.

[17] A. Nandi, A. Ganjam, P. Druschel, T. E. Ng, I. Stoica, H. Zhang, and B. Bhattacharjee, "SAAR: A shared control plane for overlay multicast," in *Proc. 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Cambridge, MA, USA, April 11-13 2007.

[18] "Wireshark," http://www.wireshark.org.

[19] F. Wang, J. Liu, and Y. Xiong, "Stable peers: Existence, importance, and application in peer-to-peer live video streaming," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, Phoenix, Arizona, April 14-17 2008.

[20] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live P2P streaming approaches," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, Anchorage, Alaska, May 2007.

[21] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?" *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1678–1694, Dec. 2007.

[22] V. Fodor and G. Dan, "Resilience in live peer-to-peer streaming," *IEEE Communications Magazine*, vol. 45, no. 6, pp. 116–123, June 2007.