

Mapping the PPLive Network: Studying the Impacts of Media Streaming on P2P Overlays

Long Vu, Indranil Gupta, Jin Liang, and Klara Nahrstedt

Department of Computer Science

University of Illinois at Urbana-Champaign, IL 61801

Email: {longvu2, indy, jinliang, klara}@cs.uiuc.edu

Abstract—While several deployed p2p file sharing overlays have been characterized in the literature, this paper shows that some of their conclusions may be false for p2p applications that stream media instead. Specifically, we undertake a crawler-based investigation of PPLive, the largest live multimedia streaming system in the world today. It is important to understand IPTV overlays like PPLive in order to enable the building of larger-scale media streaming overlays. Our task is challenging because PPLive is proprietary. PPLive has multiple channels, each channel with its own overlay, and a large fraction of these channels stream preset movie schedules. A human user may join any given channel, but the user's client machine could be used to relay feeds for other non-subscribed channels too. Popular PPLive channels contain several thousands of nodes.

We crawl the real deployed PPLive network via both machines in a cluster at UIUC, and by using PlanetLab hosts. Our major findings are: (1) Unlike p2p file sharing users, PPLive peers are impatient, (2) Channel Size variations are larger than in p2p file sharing networks, (3) Average degree of a peer in the overlay (i.e., its out-degree) is independent of channel size, (4) Smaller PPLive overlays are similar to random graphs in structure, (5) The availability correlation between PPLive peer pairs is bimodal, i.e., some pairs have highly correlated availability, while others have no correlation. We believe these results point us towards taking seriously the nature of applications while designing and optimizing p2p overlays.

I. INTRODUCTION

The proliferation of large-scale peer-to-peer (p2p) overlays such as Kazaa, Gnutella, Skype [1], PPLive [2], RONS [3], etc., has created the need to characterize, and to understand the emergent properties of these overlays. A large fraction of existing characteristic studies focus on *file-sharing* p2p applications, such as Kazaa, Gnutella, and Napster. Some of the more prominent studies among these are by Ripeanu et. al. [4] on Gnutella, by Saroui et. al. on Napster and Gnutella [5], and by Bhagwan et. al. on Overnet [6]. Although these studies have created a better understanding of the characteristics of p2p overlays, there is a risk that many systems designers may believe that some of these conclusions are in fact shared by many other p2p overlays.

This paper shows that many of the well-held beliefs about the characteristics of p2p file sharing overlays are in fact false when one changes the application atop the p2p overlay. Specifically, we undertake a crawler-based study of a deployed application overlay network called PPLive. PPLive is perhaps the most well-known instance of an IPTV (Internet Protocol

Television) application. IPTV applications have seen a dramatic rise in popularity and have received significant attention from both industry and academia. The number of subscribers is predicted to increase from 3.7 million in 2005 to 36.9 million by 2009. Revenues could reach US\$ 10 billion at the end of this period [7]. This promising market has encouraged the rapid development of IPTV technologies including tree-based multicasts [8][9][10], receiver-driven p2p streaming [11][12][13], and chunk-driven p2p streaming [2][14]. Among these IPTV approaches, chunk-driven p2p streaming has emerged the most successful technology in terms of the number of simultaneous viewers [15].

Besides PPLive, there are currently several other chunk-driven p2p streaming systems developed by different communities. One of these is CoolStreaming/DONet which was released in May 2004 and attracted over 30,000 users within a year [14]. Coolstreaming relies on a BitTorrent-like protocol [16]. Although Coolstreaming is an open-source, we choose to study PPLive in this paper because it is larger by an order of magnitude and is likely to show many more emergent properties than CoolStreaming.

PPLive is the largest chunk-driven multimedia streaming p2p overlay in the world. As of May 2006, PPLive had over 200 distinct online channels, a daily average of 400,000 aggregated users, and most of its channels had several thousands of users at their peaks [2]. The system is increasing in popularity, especially in China and Asia. For instance, during the Chinese New Year 2006 event, a particular PPLive channel had over 200,000 simultaneous viewers [15]. In our experiments during July 2006, we observed that there are about 400 online channels on the English version page of PPLive [17].

PPLive streams live TV and video data through overlays of cooperative peers. The PPLive system has multiple channels, each of which forms its own overlay. Each channel streams either live audio-video feeds, or movies according to a preset schedule. A human user may join any channel via her client machine, but the client machine could also be used to relay feeds for channels other than the subscribed one (by the PPLive protocol).

Understanding how a large deployed system like PPLive manages such a big network is essential for developing large-scale IPTV applications in the future. There are several

measurement studies about PPLive characteristics [15][18]. In these papers, authors focused on evaluating PPLive performance such as channel population, user arrivals and departures, user geographic distribution. They also evaluated video traffic, video TCP connections, and user-perceived quality. These studies concentrated on measurement rather than relationship between channel characteristics, user behavior, and overlay characteristics. Nevertheless, there is no research on *how channel properties and user preference influence the system overlay*. Our study attempts to fill this gap.

Results obtained from our experiments indicate that PPLive overlay characteristics differ from those of p2p file sharing. Our major findings are that: (1) Unlike p2p file sharing users, PPLive peers are impatient, (2) Channel Size variations are larger than in p2p file sharing networks, (3) Average degree of a peer in the overlay (i.e., its out-degree) is independent of channel size, (4) Smaller PPLive overlays are similar to random graphs in structure, (5) The availability correlation between PPLive peer pairs is bimodal, i.e., some pairs have highly correlated availability, while others have no correlation. All the above conclusions, except (3), are markedly different from the well-known characteristics of p2p file sharing systems [6][4][5].

Studying PPLive is challenging because (a) of its size and dynamism of viewing content and user population, and (b) it is a proprietary protocol with few publicly known design decisions. As a result, our study follows two principles: I. Careful design and thorough validation of our crawler; II. Careful definition of measured metrics in order to derive unbiased results and draw correct conclusions. These are explained in more detailed in Sections II and III. Our hypotheses are validated by extensive experiments with data crawled during a period of 4 months, stretching from April 2006 until the end of July 2006.

The rest of this paper is organized as follows. We start by describing PPLive basics and preliminary definitions for our study in Section II. Section III presents and justifies our crawler methodology. In Section IV, we show why PPLive peers are impatient. Section V discusses the impacts of episode-based PPLive channels on the overlay and its size. We study the structure of the PPLive overlay in Section VI and evaluate host availability interdependence in Section VII. We conclude and discuss in Section VIII.

II. PPLIVE BASICS AND PRELIMINARY DEFINITIONS

Before embarking on a characteristic study of PPLive, we briefly summarize its basic architecture as well as the structure of its channels, in each case giving some basic definitions that will be reused later in the paper.

A. PPLive Architecture

PPLive is a free IPTV application which divides video streams into chunks and distributes them via overlays of cooperative peers. The PPLive system consists of multiple overlays, with one overlay per channel. Each channel streams either live content or a repeating prefixed program, and the

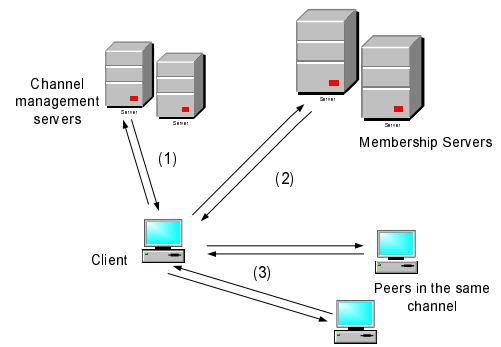


Fig. 1. PPLive membership and partnership protocols.

feed from the channel originates from a server. A user can join at most one channel. When she does so, her client machine is not only as a consumer of feeds from that channel, but may also be chosen by the protocol to act as a relay for feeds from other channels. By default, each PPLive client has a pair of *TCP* and *UDP* ports to communicate with PPLive servers and partners. A number of other *TCP* ports can be used by the client to exchange video chunks during its sessions.

The first challenge is that in PPLive, it is very difficult to distinguish between the notion of “user” and “client machine”. There are two main reasons for this: (1) PPLive users are free to join, leave, and switch channels by accessing the PPLive web interface or PPLive Net TV. (2) Due to NAT boxes and firewalls, a user’s client machine may change its *IP* or *UDP* port number or both. (3) The proprietary PPLive system is rumored to use the idea of inter-overlay optimizations [19]; as a result, a client machine may appear as a participant in multiple overlays, including ones that the user is not subscribed to.

Hence, in the rest of this paper, we refer to a given $\langle IP, port \rangle$ tuple as a “node” or a “peer” - this is a combination of both a client machine and a user. The term “client” refers only to the machine (e.g., workstation) that the PPLive node is running on, while “user” refers to the human user, and these should not be confused with node or peer.

1) *PPLive Protocols*: Although PPLive is not open-source, a little of its internal design decisions are known. Each PPLive node executes two protocols, for (1) registration and harvesting of partners, and (2) p2p video distribution. A PPLive node maintains two kinds of partners: “candidates” and “real” partners. The latter type are used for exchanging video streams via *TCP* connections, while the former is used to replace real partners that have become unresponsive. For our study, we use a capability where a node can be queried for its candidate and real partner lists, and it returns this *partner list* in a message. One difficulty is that it is not known whether this returned list is the full partner list, or a subset of it. Hence, we need to define a notion of “node degree” and “partner list” that is generic and covers both possibilities - we will do so soon.

Figure 1 shows the actions of a PPLive node to join the network: (1) retrieve a list of channels from channel management servers via *HTTP*; (2) for the interested overlay,

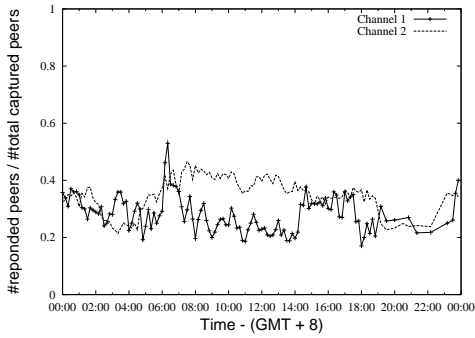


Fig. 2. Percentage of #responded peers and #total captured peers. Over 50% peers do not respond to probing message.

retrieve a small set of member nodes from the membership servers via *UDP*; (3) use this seed partner list to harvest (learn about) other candidate partners by periodically probing existing partners (and sometimes membership servers) via *UDP*. If a PPLive node is inside *NAT* or firewalls, *UDP* in the above steps may be replaced by *TCP* - further details are in [15].

2) *PPLive Overlay*: We formally define the PPLive overlay as a graph $G = (V, E)$. Recall that each PPLive overlay corresponds to an individual PPLive channel. Each node (or peer) is defined as a given $\langle IP, port \rangle$ tuple and belongs to V . Each partner (or neighbor) of this node, appearing in its partner list, then corresponds to an edge (or link) in E .

k response degree: We are interested only in the out-degree of a node in the overlay, and henceforth call this simply as “degree”. As discussed previously, we need to define degree in a manner that is generic. The *k response degree of a node* is defined as the aggregated set of partners returned in the first k responses from a node that is repeatedly (once a second) sent a query for its partner list. We use a default setting of $k = 15$, however we verify the generality of these results for smaller values of k as well (Section VI-B). Henceforth in this paper, the term *node degree* stands for *k response degree*.

3) *Active Peer*: The next challenge is to clearly define when a peer is considered to be a part of a given overlay. This is complicated by the fact that a user may not be subscribed to a channel that the user’s client machine is participating in. Further, some clients may be behind NATs or firewalls, and may not respond to a direct probe message.

Given an overlay G and a peer v , v is considered to be an *active peer* in G if either v appears in the membership list for G at one of the membership servers, or v is present in the partner list of some other peer u that is also an active peer. Notice that the definition is recursive. Formally, we define the predicate:

$$ACTIVE(v, G) = \{v \in \text{Membership Server List for } G\} \text{ OR } \{\exists u : ACTIVE(u, G) \text{ AND } v \in u.\text{PartnerList}(G)\}$$

This definition is more inclusive than that in [15] because our definition also includes “silent” peers that may be behind firewalls. Even though we have not described our crawler yet, we need to justify the above definition. We quickly present

| Catalog Name | Number of channels |
|---------------|--------------------|
| TV | 52 |
| Information | 29 |
| Sports | 1 |
| PhonenixTV | 5 |
| Movies | 79 |
| Teleplay | 66 |
| Entertainment | 68 |
| Cartoon | 30 |
| Game | 28 |
| Others | 52 |
| Summary | 410 |

TABLE I
PPLIVE CHANNEL TYPES.

two simple experiments below to do so.

First, we measured the fraction of peers that were captured by our crawler (see *Snapshot Operation* in Section III) using the above definition, but that did not respond to a direct ping. Figure 2 shows the fractions for two different PPLive channels over the course of 24 hours. The authors of [15] reported that around 40% nodes may be behind NATs. Since Figure 2 shows that over 50% of the captured peers are non-responsive: it is important to consider the characteristics of these peers as a part of the overlay, and our definition does this.

Secondly, we verify that the p2p membership services of the PPLive protocol do in fact spread updates rather quickly. We chose a large channel, and ran 10 globally-distributed instances of our crawler, each at a randomly chosen PlanetLab node. Under this, we first had a PPLive node join a particular channel and ran the crawlers 15 seconds after the join. All the 10 crawlers captured the new node. Then we killed the PPLive node, and 15 seconds later, ran the crawlers again. None of the crawlers returned the node in its membership list. Thus we are reasonably confident that the p2p membership services of the PPLive protocol spread membership updates quite rapidly.

Finally, for our crawler, we define the *channel size* as the number of active peers attending the channel in a certain period of time, usually one execution of the crawler. We use channel size interchangeably with “channel population”, and “overlay size.”

B. PPLive channels

PPLive Channels are also organized into catalogs such as TV, Movies, Entertainment, Game, etc. Table I shows catalogs with corresponding number of channels. The PPLive home page reports over 200 simultaneous daily channels. During our experiments in July 2006, we observed over 400 channels. PPLive channels fall into two categories: (1) live (TV) channels, and (2) episode-based channels. Our study considers only the latter category, and we justify this choice below.

1) *TV channels*: This group consists of channels which display live content taken from a real TV station. Viewing contents of these channels vary on a daily basis. This group corresponds to the first row in Table I.

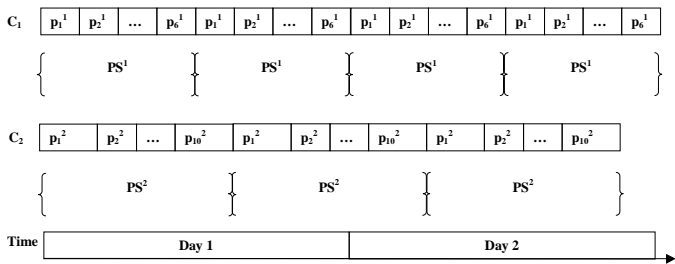


Fig. 3. Channel C_1 and C_2 with corresponding program set PS^1 and PS^2 over 2 consecutive days. PS^1 has 6 programs with the same length, p_j^1 is 2 hours. Meanwhile, PS^2 has 10 programs with different lengths. In 24 hours, C_1 streams two PS^1 s while C_2 streams one PS^2 .

2) *Episode-based channels*: An episode-based channel has a fixed preset program set (usually a fixed set of movies in a given order), and repeats this program periodically. The length of the periodic program set can vary from a few hours to around 25 hours. Formally, for a given channel i (e.g., the “Jackie Chan Movie channel,” or the “Mouse and Cat cartoon channel,” etc.), let a program set PS^i be defined as follows:

$$PS^i = \{p_1^i, p_2^i, \dots, p_n^i\} \quad (1)$$

where each p_j^i is a program (usually distinct from p_k^i , for $k \neq j$), such as a movie (e.g., Jackie Chan’s movie “Who am I?”), or a cartoon strip, etc. Then the program on the episode-based channel C_i is the following repeating chronological sequence:

$$C_i = \{PS^i, PS^i, \dots, PS^i\} \quad (2)$$

Channels schedule their program sets independently of other channels. Figure 3 illustrates the timeline of two example channels C_1 and C_2 , over a period of 2 consecutive days. C_1 ’s program set PS^1 contains 6 equi-sized programs with a total length of 12 hours, while C_2 ’s program set PS^2 contains 10 different-sized programs including with a total length of 15 hours.

In this paper, we focus only on episode-based channels because of following reasons. First, Table I indicates that TV channels accounts for only one-eighth of all channels. Second, episode based channels are larger because they can nurture and maintain a devoted fan following. For instance, the PPLive homepage advertises daily a list of 20 “recommended” channels rated on viewing quality and popularity. A large fraction (90% to 100%) of these advertised channels are episode-based. Thus, episode-based channels are the most significant contributor to the PPLive population and network bandwidth, etc.

III. STUDY METHODOLOGY

In this section, we first describe our experiment settings. We then list basic metrics we are interested in measuring, and describe our two crawler-based operations: “Snapshot Operation” and “Partner Discovery.” Notice that “snapshot” in our case is quite different from the classical notion of Chandy and Lamport snapshots.

TABLE II
THE THREE REPRESENTATIVE PPLIVE CHANNELS INVESTIGATED IN OUR EXPERIMENTS.

| Name | Channel size | PS Len | #Progs | Prog Len | Type |
|------|--------------|--------------|--------|--------------|---------|
| A | 35000-45000 | 6h15min | 6 | 36min-2hours | Movie |
| B | 8000-12000 | 4day 4h | 300 | 20min | Cartoon |
| C | 10000-15000 | 1day 2h16min | 40 | 40min | Movie |

Experiment settings: Our crawler works in the following manner: a machine (either a Linux machine in our CSIL cluster at UIUC, or a PlanetLab node) joins a given PPLive channel, and then crawls it. The crawler is used to execute what we call the *Snapshot Operation* (not to be confused with the classical notion of Chandy and Lamport snapshots [20]), which will be described shortly. For the PlanetLab setting, we crawl using 10 geographically-distributed hosts, and aggregate their crawled information. Ethereal [21] is used to trace traffic between PPLive nodes and servers.

Most of our experiments were focussed around three specific representative channels from PPLive - Table II shows their basic features. We anonymize these channels by naming them as A, B, and C. A is the most popular channel, B has shorter programs and the largest program set, while C is somewhat in between A and B. Since a large fraction of PPLive users are in China, we use Chinese Time Zone (GMT+8) in our plots.

Snapshot Operation: We are interested in measuring the following metrics: (1) session length of peers, (2) channel size, (3) peer degree, (4) overlay clustering coefficient, and (5) availability correlations. In order to support this, we use our crawler to develop a *Snapshot Operation*. This “Snapshot” is different from the notion of Chandy and Lamport’s “consistent snapshots” [20].

Intuitively, the Snapshot operation for a PPLive overlay attempts to capture the set of nodes present in the overlay over a short period of time ($O(\text{minutes})$), along with their partner lists. It works by repeatedly fetching partner lists and querying returned entries. Specifically, the snapshot operation works as follows:

- 1) The initiator (either our CSIL node or one of our PlanetLab nodes) first requests the initial peer list from one of the PPLive membership servers, and uses this to initialize a local round-robin list denoted as L .
- 2) The initiator then continuously scans the list L in a round-robin fashion, by sending a request for partner list to each entry, and appending to L new peers (i.e., ones that it has not heard about before) received in the partner list replies.
- 3) The Snapshot operation terminates when the initiator has received fewer than k new peers among the last Δ peers received as partner lists. We use $k = 8, \Delta = 1000$ in all of our experiments; with this setting, the snapshot operation typically takes between 3 to 8 minutes. To avoid flooding network with our messages, new snapshot operations are initiated only once every 10 minutes.

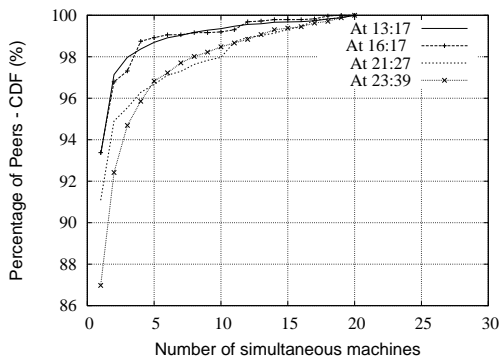


Fig. 4. 10 machines simultaneously running the snapshot operation can crawl 98% peers in one channel, compared to when 20 machines are used. We use 10 Planetlab nodes in different locations around the world to run the snapshot operation in our experiments.

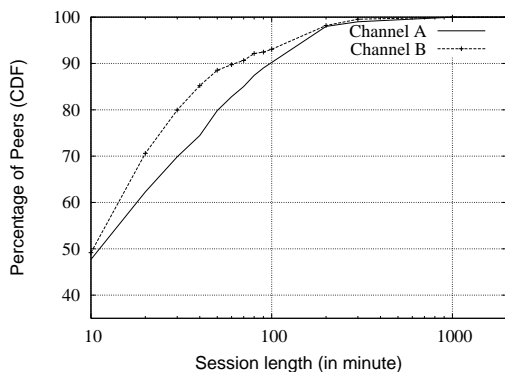


Fig. 5. PPLive peers are impatient. CDF of session lengths is shown. The X-axis is on a log-scale.

The above snapshot operation is different from the crawler of [15] in two ways – their crawler runs once each minute and for about 15 seconds; thus to crawl a large part of the network, it imposes a high load on PPLive. Second, their crawler counts only responding peers, thus undercounting the channel size, i.e., they use a more restrictive definition for $ACTIVE(u)$ than our definition in Section III.A.

To increase the coverage of our snapshot, we run it in parallel on multiple machines. Figure 4 shows the captured number of peers with k machines as a fraction of the captured number of peers with 20 machines (at four different times). We observe that 10 machines cover about 98% of peers covered by 20 machines. Hence, in this paper, we decided to use 10 PlanetLab nodes to run simultaneous snapshot operations.

Partner Discovery: The operation to obtain the k response degree of a node (Section II-A.2), we repeatedly request a peer (approximately once every second) to send its partner list. The first k received responses are aggregated to create the k response degree (and k response partner list).

IV. PPLIVE PEERS ARE IMPATIENT

With the basics and study methodology discussed in the last two sections as building blocks, this section discusses our first result. It has been widely reported, e.g., [5], that users of

p2p file sharing systems are “patient,” i.e., they do not mind waiting hours, and sometimes even days, for a file as large as 1 GB, to download.

In the PPLive environment, due to the streaming nature of the content, the opposite is true. Using our snapshot operation (Section III), we define the session length of a PPLive node as the time covered by consecutive snapshots that all contain this node (i.e., 10 minutes times the number of consecutive snapshots). Notice that this session length is an *overestimate* of the real session lengths (since the given peer may be online for less than a multiple of 10 minutes).

Figure 5 shows session lengths of 5000 random peers taken from 38675 peers in channel A, and 5000 random peers taken from 11625 peers in channel B. We observe that about 50% sessions are shorter than 10 minutes, 60% of A’s sessions and 70% of B’s sessions are shorter than 20 minutes, and over 90% sessions from both channels are 100 minutes or shorter. This implies that *PPLive nodes are impatient*, i.e., they rarely stick to a channel for too long.

This opposite behavior arises out of both a difference in application characteristics, as well as from user behavior. Since p2p file sharing overlays like Kazaa are *batch-mode* delivery systems in which the human users can go away from the client machine while it continues to download content, session lengths are long. In comparison, the PPLive application is a *streaming-mode* one, where the user can obtain utility from the application only if she is actively present near the client machine. If the user is not sitting at her machine, she has no incentive to keep PPLive running, hence the session times are shorter.

There are other reasons (both application and user-based) contributing to the short session lengths. First, PPLive users are likely to switch from one channel to another because of a loss of interest - home television viewing often suffers from the same malady! Secondly, PPLive nodes face a longer start-up delay than nodes in p2p file sharing systems or traditional TV. We have observed that newly joining nodes need tens of seconds to a minute to join a channel, with the latency being even higher if the channel is really small (due to the scarcity of potential neighbors). Furthermore, the long start-up delays increase the likelihood of the user switching to a different (and more popular) channel.

The difference in session length distributions for channels A and B in Figure 5 also brings out an important observation – *short-term sessions depend on channel characteristics, while long-term sessions depend on user preference*. For short term behavior – notice that 60% of channel A’s sessions are shorter than 20 minutes, while as many as 70% of the channel B’s sessions are shorter than 20 minutes. This behavior arises because channel A is more popular, as well as because channel B’s programs are shorter (around 20 minutes) compared to channel A. In contrast, long-term sessions of the two channels converge, indicating that long-term sessions depend on user preference – for any channel, about the same fraction of users stays “too long.”

In conclusion, the application characteristics and user be-

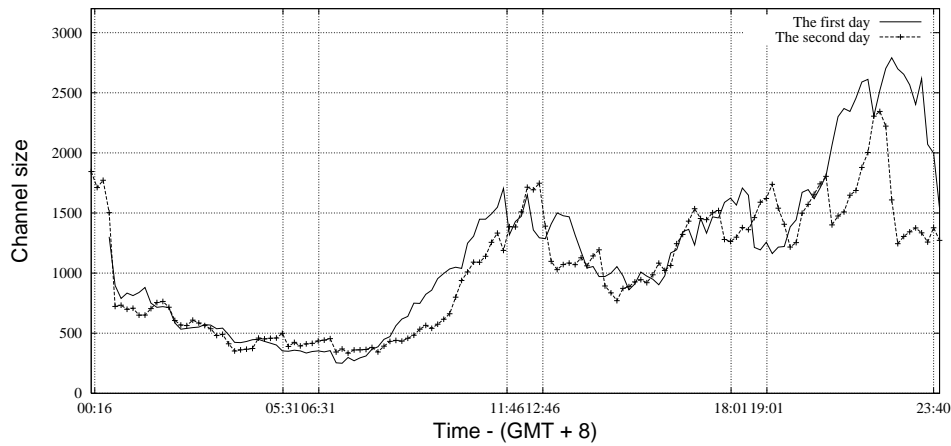


Fig. 7. Channel size distributions over 2 consecutive days. Starting times of *PS*s on the first day are 05:31, 11:46, 18:01. Starting times of *PS*s on the second days are 00:16, 06:31, 12:06, 19:01. The *PS* length is 6 hours, 15 minutes, hence the *PS* drifts forward 1 hour from the first to the second day. However, the channel size peaks drift too.

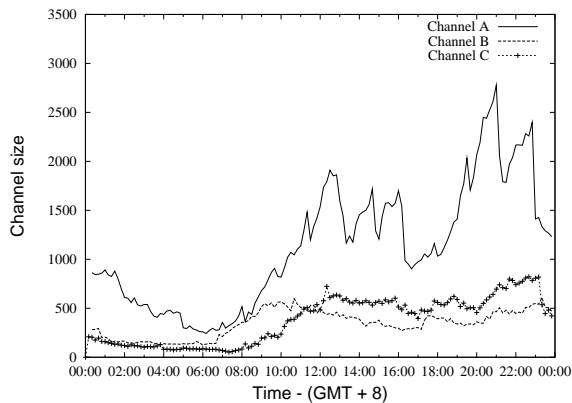


Fig. 6. Channel size distribution. Channel size peaks at noon and night, and is smallest in the morning. Since B and C have equal length programs, their channel sizes are low, stable, and similar to each other. Meanwhile, A's channel size varies significantly with time as its programs have differing popularities and program lengths.

haviors cause *very short session lengths and consequently a higher degree of churn in PPLive than in p2p file sharing overlays*. Future designs for both streaming p2p overlays and “generic p2p routing substrates” will have to keep these in mind.

V. CHANNEL SIZE VARIES WIDELY OVER A DAY

Studies on file sharing p2p systems [6] showed that diurnal patterns and churn exist, but the size of a p2p overlay stays stable in spite of these features. The findings in this section show that PPLive-style networks have highly variable channel size (as well as high churn and diurnal patterns).

This section studies the time variation of channel size in PPLive channels. We conduct two experiments - the first compares three different channels, while the second studies a single channel over a longer time.

For the first experiment, Figure 6 shows the variation of channel size for the three PPLive channels over the course of

a day. We observe that all channels have peak populations at noon and evening/night, and are smallest in the morning. This is clearly due to the influence of time-of-day.

Next, notice that the distribution of channel sizes for B and C look much flatter than A – the former two vary between 100 and 600, while A shows a variance of over 2000. This arises from the content of these channels. B and C are series of equal length programs while A consists of longer movies. As a consequence, compared to A, people join channel B to watch short skits and leave more quickly, while viewers that join A stay on for longer, perhaps long enough for the movie to end. Finally, notice that the channel size peaks are both unevenly spaced and are different in height - this is because A's programs have variable lengths (and popularities).

In the second experiment, we crawl peers in channel A for two consecutive days. As presented in table II, *PS* of channel A is 6 hours and 15 minutes, thus there is a drift forward of 1 hour from one day to the next. Figure 7 shows the distribution of channel size over these two days. Notice that the peak around noon-time on day 2 occurs one hour later than the corresponding peak on day 1. The same observation holds for the peak at evening time (18:01 pm on day 1) and the peak at night time (21:00 on day 1).

Observing each program in Figure 7 (e.g., the segment from 05:31 to 11:46 on day 1) also reveals that each program has a similar behavior trend – the channel size begins to fall once the program starts, bottoms around the mid-point, but then picks up and peaks at a higher point than the size at the program start. This is likely because a large number of users join the channel to only watch the final part of the program (movie).

Finally, notice that the peaking behavior is influenced not only by the nature of the program but also by the time-of-day: the peak at night time (21:00 on day 1) occurs towards the middle of a program; this is different from the noon-time and evening peaks, both of which occur towards the end of a program.

In conclusion, unlike p2p file sharing systems which only

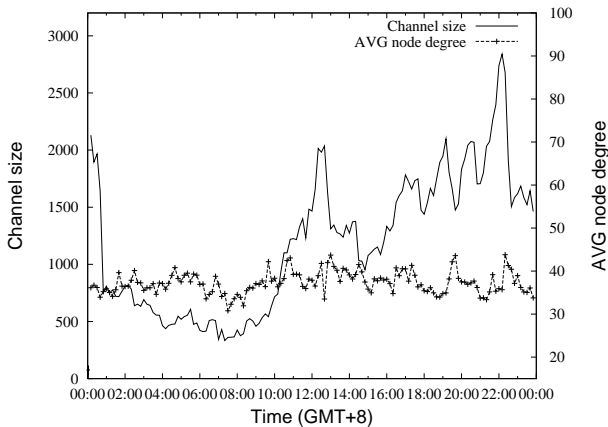


Fig. 8. Average node degree is independent of channel size.

show diurnal patterns, p2p media streaming environments are more volatile (show higher variation in channel size), and the channel size is influenced heavily by the popularity of the content being broadcasted, as well as by time-of-day. On the positive side, the channel size variation is somewhat self-repeating and perhaps even predictable (crests and troughs occur at predictable times). It may be possible to design “channel size predictors” for such systems, which would address the churn problem more directly than possible in other p2p systems.

VI. PPLIVE OVERLAY STRUCTURE MAY BE SIMILAR TO RANDOM GRAPHS

It is well-known that the degree distribution in file sharing p2p networks is scale-free and hence likely small world [5][4]. This section shows that like file sharing p2p overlays, the average node degree in the PPLive overlay is also independent of the channel size. However, in direct contradiction to file sharing p2p overlays, the structure of the PPLive overlay is closer to that of random graphs, under certain situations.

A. Average node degree is independent of channel size

We collect the k response partner lists of nodes using our Snapshot Operation and Partner Discovery engine (Section III). During a given snapshot operation (over 10 minutes), we simultaneously run the Partner Discovery to obtain the k response partner lists of 200 randomly selected peers that are both active and responsive. Figure 8 shows the variation of the average k response node degree (henceforth degree) of channel A during a 24 hour period, for $k = 15$.

These figures first indicate that although the average node degree varies, it stays within a small range - between 30 to 43 over the course of day. More importantly though, *there appears to be no correlation between the variation of average degree and the channel size*. Thus we conclude that the average degree of a PPLive node does not depend on the channel size.

This behavior can be explained since a peer in an overlay only needs to communicate once in a while with a few other of peers to exchange video data, advertise availability, and

discover new partners. Therefore, even when the channel becomes large, a peer can still preserve viewing quality without establishing too many new *TCP* connections.

In conclusion, this is one of the few characteristics of the PPLive overlay that is shared with file sharing p2p overlays. “Scale-free” file sharing p2p overlays show a constant node degree independent of overlay size too [4].

B. Randomness of overlay depends on channel size

However, many file sharing p2p overlays are reputed to be small-world in nature. In contrast, the PPLive overlay may, under some conditions, resemble a random graph.

The distinction between a random and a non-random graph can be measured by a metric called “Clustering Coefficient” (CC). Watts and Strogatz first used CC to characterize graphs in [22]. Informally, the CC in a graph is defined as follows: for a random node u with two neighbors v and w in its partner list, the CC is the probability that either v is in w ’s partner list, or vice versa. For a random graph, the value of CC will be closer to the unconditional probability (i.e., even if v and w were not neighbors of u) that either v is w ’s neighbor, or vice-versa. The farther the CC is from the above value, the less random (i.e., more clustered) the graph is.

For our experiment, we first calculate the average degree of the PPLive overlay (measured as in Section VI-B), and use it to calculate D as follows:

$$D = (\text{average node degree}) / (\text{Channel size}) \quad (3)$$

We then compare the CC (measured as described below) to D (the unconditional probability that v links to w). The CC information is measured simultaneously to the degree measurement, and as follows: in each snapshot, we randomly select 200 active peers. Call each such chosen peer as a root node R , we first use partner discovery to obtain its partner list. Second, we pick randomly two active partners P_1 and P_2 in R ’s partner list and obtain their partner lists, again via the partner discovery operation. Notice that both these partner discovery operations use the notion of k response partner lists. Third, we verify whether P_1 is in P_2 ’s partner list or not, and vice versa. If P_1 is in P_2 ’s partner list (or vice versa), we increase a variable called *Count* by 1. *Count*, initialized to 1, represents the total number of edges existing in all such partner pairs. Then, *CC* is computed as follows:

$$CC = \text{Count} / (2 * \text{RootNodeNum}) \quad (4)$$

The above CC should be close to the value of D in a graph that is random. Figure 9 plots, for two different values of $k = 5, 15$, the 24-hour variation of D and *CC* for channel A. This experiment was done at the same time as Figure 8. Thus, notice that from 04:00 am to 08:00 am, when the channel size is small, the value of *CC* approaches the value of D , especially for higher values of k . This indicates that *when channel size is small, the structure of the PPLive overlay graph is not small-world, but instead approaches a random graph*.

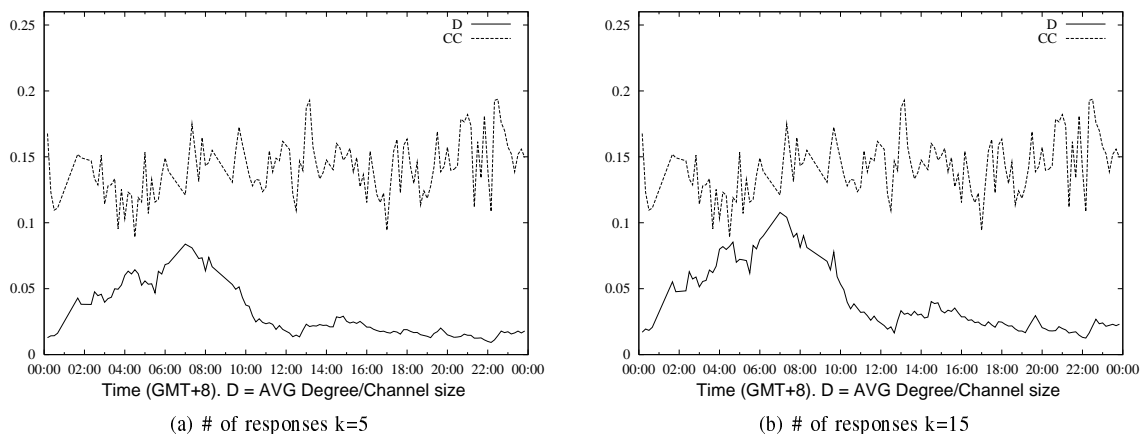


Fig. 9. Overlay resembles a random graph when channel size is small (around 500 nodes) but becomes more clustered when channel size grows. Relationship between D and CC with different k values. Different k values have similar shape.

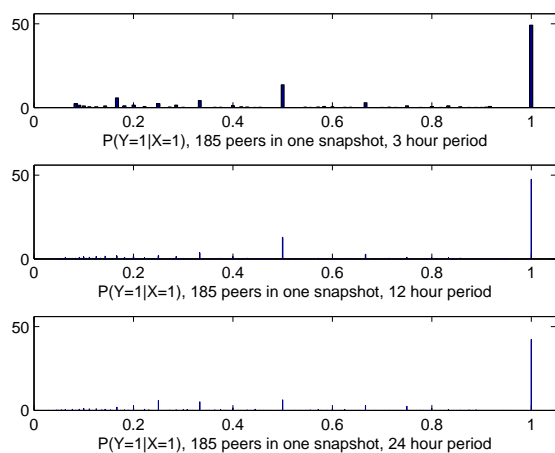


Fig. 10. Peers occurring in the same snapshot may occur together again. PDF of availability correlation. Plot shows data for all peer pairs among 185 peers in one snapshot of channel A. Y-axis is % host pairs.

This is explained by the fact that for “small” channels (with as many as 500 nodes), peers indeed connect fairly randomly to each other. As the channel size increases (10:00 am onwards in Figure 9), the CC is only about six times that of the value of D . This is still indicative of some randomness of the graph, although it is clear large channel sizes lead to more clustering.

In conclusion, while file sharing p2p systems are known to exhibit small-world behavior, media streaming p2p systems are less small-world, especially at “small” channel sizes (with as many as 500 nodes). It is unclear whether this behavior is due to the PPLive protocol or from the application’s influence - a deeper look at PPLive’s internal design may resolve this.

VII. PEER AVAILABILITY INTERDEPENDENCE

File sharing p2p systems are known to have host availabilities uncorrelated [6]. In comparison, we show that: (1) unlike in p2p file sharing systems, PPLive peer pairs occurring together in some snapshot have highly *correlated* availabilities,

while (2) like in p2p file sharing systems, peer pairs that are randomly selected will have highly *uncorrelated* availabilities.

We measure the correlation between the availability of two peers X and Y by using a similar technique as in [6]. Specifically, let $X = 1$ (resp. $Y = 1$) be the event that peer X (resp. Y) occurs as an active peer in a given snapshot. Then, for the peer pair (X, Y) , we calculate $P(Y = 1|X = 1)$, i.e., the conditional probability that given X is present in a given snapshot, Y will be too. We then compare this conditional probability to the unconditional probability that peer Y occurs in a given snapshot, i.e., $P(Y = 1)$. The closer the two values, the more uncorrelated are X ’s and Y ’s availability patterns.

A. Nodes in the same snapshot have correlated availability

Given traces of a series of snapshots (for Channel A) taken over a contiguous time period (we use three settings: 3 hours, 12 hours, and 24 hours), we select a set of 185 peers from one particular snapshot (we use the smallest snapshot). Figure 10 shows the conditional probability $P(Y = 1|X = 1)$, for each node pair in this set. 50% of node pairs show a high correlation in availability, i.e., $P(Y = 1|X = 1) = 1$.

We believe there are two factors contributing to this behavior: first, user pairs that appeared in the same snapshots are likely to have similar interests in terms of channel viewing contents. Secondly, and perhaps more importantly, certain peer pairs that occur together in a snapshot are perhaps “well-matched” as streaming relays for each other. It is possible that PPLive’s inter-overlay optimizations [19] cause one client’s presence to draw in other well-matched clients for relaying.

B. Random node pairs have independent availabilities

We ran a similar experiment as in Section VII-A, except that we selected 500 random peers from among 39412 crawled peers of channel A, as well as 500 random peers from 11527 peers of channel B, each collected over a 24 hour period. Then, we computed the difference between $P(Y = 1|X = 1)$ and $P(Y = 1)$ for each host pair (among the set of 500) over the 144 snapshots, corresponding to 24 hours. In contrast to

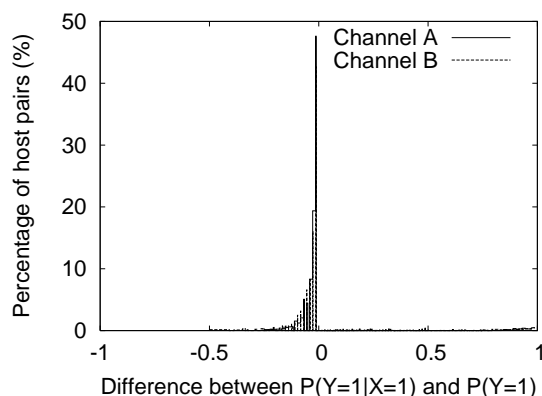


Fig. 11. Randomly selected pairs of peers have uncorrelated availabilities. PDF of availability correlation. Experiment is conducted with 500 random peers taken from channel A and channel B.

results in Section VII-A, Figure 11 shows that random peer pairs have completely independent availability behavior. In particular, 87% peer pairs in channel B (92% in channel A) lie between +0.2 and -0.2, indicating independence in availability among these peers. This is explainable because random peers are unlikely to have either correlation in user interest, or be well-matched in relaying feeds.

In conclusion, unlike p2p file sharing systems, media streaming p2p systems may exhibit a highly correlation availability among certain node pairs. Systems designers will have to account for this, regardless of whether it arises from user interests or from internal optimized PPLive design (in the latter case it is a good p2p system design principle).

VIII. CONCLUSIONS AND LESSONS LEARNED

While several characteristic studies exist for p2p file sharing overlays, many well-known conclusions for those overlays become false when one considers p2p applications that stream media atop the overlay instead. In this paper, we studied the overlay characteristics of the proprietary PPLive protocol, currently the world's largest and most popular p2p media streaming system. This study is also valuable since IPTV applications are growing rapidly, yielding new classes of overlays. Our data was collected during April-July 2006. Using a lightweight snapshot operation and partner discovery, and generic metrics (e.g., k response degree), we crawled several PPLive overlays and "nodes" within. A node is the combination of a human user and her client machine. Our studies found that PPLive overlay properties are heavily influenced by both user behavior and application nature.

Specifically, we found that: (1) PPLive nodes are much more impatient than p2p file sharing users, thus indicating a higher rate of churn. (2) PPLive channel sizes (especially for popular channels) vary much more than in p2p file sharing systems, yet the channel size variation has repetitive patterns due to the episode-based channels. (3) Like in p2p file sharing overlays, the average degree of a PPLive node is independent of the channel size. (4) Unlike p2p file sharing overlays,

small PPLive overlays (channels) resemble random graphs in structure, while large PPLive overlays retain some of this "randomness" too. (5) PPLive node pairs have a bimodal distribution in their availability correlation - node pairs that occur together in one snapshot are likely to occur together again with high probability, while node pairs selected at random have uncorrelated availabilities.

These differences between PPLive overlays and p2p file sharing overlays such as Kazaa, Gnutella, Napster, etc., show that p2p systems designers may need to account for application nature. This study is also indicative of the challenge in designing "generic" p2p substrates catering to a wide variety of applications. Since custom-built substrates are too wasteful, it may thus be important for systems designers to address classes of p2p applications with common characteristics. Finally, a deeper study of user behavior (e.g., via HCI research) may yield novel p2p overlay design principles.

REFERENCES

- [1] (2006) Skype homepage. [Online]. Available: <http://www.skype.com/>
- [2] PPLive homepage. [Online]. Available: <http://www.pplive.com/>
- [3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *ACM SOSP*, 2001, pp. 131-145.
- [4] M. Ripeanu, I. Foster, and A. Iamnitchi, "'Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design,'" *IEEE Internet Computing Journal*, vol. 6, no. 1, 2002.
- [5] Saroiu, P. K. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of Napster and Gnutella hosts," *Multimedia systems*, 2003.
- [6] R. Bhagwan, S. Savage, and G. Voelker, "Understanding availability," in *IPTPS*, 2003.
- [7] Multimedia research group inc. [Online]. Available: http://www.mrgco.com/TOC_Global_Forecast_0805.html
- [8] Y. hua Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *ACM SIG-METRICS*, 2000.
- [9] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *ACM SIGCOMM*, 2002.
- [10] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *IEEE Infocom*, 2003.
- [11] M. Hefeeda, A. Habib, D. Xu, B. Bhargava, and B. Botev, "Collectcast: A peer-to-peer receiver-driven overlays," in *ACM Multimedia*, 2003.
- [12] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Donet: A data-driven overlay network for efficient live media streaming," *IEEE Infocom*, 2005.
- [13] R. Rejaie and S. Stafford, "A framework for architecting peer-to-peer receiver-driven overlays," in *ACM NOSSDAV*, 2004, pp. 42 - 47.
- [14] 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 *IEEE Infocom*, 2005.
- [15] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," Polytechnic U., Tech. Rep., 2006.
- [16] (2006) BitTorrent homepage. [Online]. Available: <http://www.bittorrent.com>
- [17] PPLive homepage, English version. [Online]. Available: <http://www.pplive.com/en/>
- [18] X.Hei, C.Liang, J. Liang, Y. Liu, and K. W. Ross, "Insight into PPLive: Measurement study of a large scale P2P IPTV system," in *WWW 06 Wshop. IPTV Svcs. over WWW*, 2006.
- [19] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "Anysee: Peer-to-peer live streaming," in *IEEE Infocom*, 2006.
- [20] M. Chandy and L. Lamport, "Distributed snapshots: Determining global states of distributed systems," *ACM Transactions on Computer Systems*, pp. 131-145, 1985.
- [21] (2006) Ethereum homepage. [Online]. Available: <http://www.ethereal.com/>
- [22] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, 1998.