View-Upload Decoupling: A Redesign of Multi-Channel P2P Video Systems

Di Wu[†], Chao Liang[‡], Yong Liu[‡], and Keith Ross[†]
[†]Computer and Information Science [‡]Electrical & Computer Engineering
Polytechnic Institute of NYU, Brooklyn, NY, USA 11201
Email: dwu@poly.edu,cliang@photon.poly.edu, yongliu@poly.edu, ross@poly.edu

***Abstract*—In current multi-channel live P2P video systems, there are several fundamental performance problems including exceedingly-large channel switching delays, long playback lags, and poor performance for less popular channels. These performance problems primarily stem from two intrinsic characteristics of multi-channel P2P video systems: channel churn and channel-resource imbalance. In this paper, we propose a radically different cross-channel P2P streaming framework, called View-Upload Decoupling (VUD). VUD strictly decouples peer downloading from uploading, bringing stability to multichannel systems and enabling cross-channel resource sharing. We propose a set of peer assignment and bandwidth allocation algorithms to properly provision bandwidth among channels, and introduce substream swarming to reduce the bandwidth overhead. We evaluate the performance of VUD via extensive simulations as well with a PlanetLab implementation. Our simulation and PlanetLab results show that VUD is resilient to channel churn, and achieves lower switching delay and better streaming quality. In particular, the streaming quality of small channels is greatly improved.**

I. Introduction

In recent years there have been several large-scale industrial deployments of P2P live video systems, including Coolstreaming [1], PPLive[2], and Sopcast [3], etc. Recent measurement studies have verified that hundreds of thousands of users can be simultaneously participating in these systems [4], [5]. Almost all live P2P video systems offer multiple channels. PPLive and its competitors each have over 100 channels; future user-generate systems will likely have thousands if not millions of live channels.

A. Isolated-Channel P2P Video Systems

A common practice in P2P video today is to organize peers viewing the same channel into a swarm, with peers in the same swarm redistributing video chunks exclusively to each other. We refer to such a design as "*isolated-channel*" P2P video systems. Recent studies of PPLive have identified several fundamental performance problems for isolated-channel systems:

Channel switching delay and playback lag. Measurement studies of PPLive and other live P2P streaming systems [4], [5] indicate that current channel-switching delays are typically on the order of 10-60 seconds. This is clearly undesirable, as users are accustomed to delays of under 3 seconds in current cable and satellite television systems and sub-second delays when switching pages on the Web. Furthermore, these measurement studies have shown that the playback lag, from when a live video frame is emitted from the source until it is played at the peer, wildly varies from one peer to another, with delays ranging from 5 to 60 seconds. Unfortunately, the BitTorrent-like mesh-pull architectures – currently used by most of the P2P video deployments – are inherently delay and lag prone.

Poor small-channel performance. In the upcoming years, we expect to see the emergence of *user-generated live channels*, for which any user can create its own temporary live video channel from a webcam or a hand-held wireless device. Similar to YouTube, the live channel could be a professor's lecture, a little-league baseball game, a wedding, an artistic performance, or a political demonstration. In the future, there will be, at any one time, thousands of such small channels, each emanating from a relatively low-speed connection (for example, wireless PDA) and each with 10-1000 viewing peers. Measurement work [4], [5] has revealed, however, that current P2P live streaming systems generally provide inconsistent and poor performance to channels with a small number of peers.

These performance problems primarily stem from two intrinsic characteristics of multi-channel P2P video systems: *channel churn* and *channel-resource imbalance*. In multi-channel P2P video systems, churn occurs on two different time scales: peers enter and leave the video application on long time scales; and peers change channels on short time scales. A recent study of a cable television system showed that users switch channels frequently [6]. Unfortunately, this channel churn brings enormous instability to the system. For example, when a peer switches from channel A to channel B, it stops uploading to its neighbors in swarm A. Those neighbors have to find new data feed to maintain steady video inflow. And in swarm B, the newly joining peer has to find new neighbors with enough bandwidth and content to download from, leading to excessive channel switching delays.

To understand channel-resource imbalance, recall that a channel's instantaneous resource index [7], [8] is defined as

$$\rho := \frac{u_s + \sum_{i=1}^{n} u_i}{nr} \tag{1}$$

where r is the bit rate of the channel, n is the number of peers viewing the channel, u_s is the server upload capacity, and u_i is the fraction of peer i's upload capacity devoted to the channel. A channel's instantaneous resource index indicates to what

degree the upload supply matches the download demand for the channel. In particular, the streaming rate r is not sustainable if $\rho < 1$ for extended periods. In our measurement study of PPLive we observed that there is significant variation of the average resource index across channels, with some channels having $\rho >> 1$ and others in the vicinity of 1. Using PlanetLab experiments, it has been shown that streaming swarms with higher resource indexes have better streaming quality, lower startup delays and less sensitive to scheduling designs [9]. Thus, in an isolated-channel design, many channels may have poor performance simply because they are resource-index poor.

Moreover, channel switching and resource-index imbalance can conspire, leading to extremely poor performance for small channels. To see this, consider a large channel L with thousands of users and a small channel S with tens of users, and suppose both channels have an average resource index of 1.2. Due to the law of large numbers, the instantaneous resource index of channel L will almost always be greater than 1 and the swarm will provide acceptable performance. However, for channel S the instantaneous resource index will frequently dip below 1, causing highly degraded viewing quality.

In summary, the widely deployed isolated-channel designs, currently have serious performance problems, which will only become more severe as video rates increase and the number of user-generated channels grows exponentially.

B. View-Upload Decoupling

In this paper, we propose a radically different cross-channel P2P streaming framework, which we refer to as *View-Upload Decoupling (VUD)*. VUD strictly decouples what a peer uploads from what it views, bringing stability to multi-channel systems and enabling cross-channel resource sharing.

In VUD, each peer is assigned to one or more channels, with the assignments made *independently of what the peer is viewing*. For each assigned channel, the peer distributes (that is, uploads) the channel. This has the effect of creating a semi-permanent *distribution swarm* for each channel, which is formed by peers responsible for uploading that channel. This novel approach has two major advantages over isolated-channel designs:

- **Channel Churn Immunity.** VUD is *immune* to channel churn. To illustrate, suppose a peer is assigned to the swarm of channel A and is watching channel B. When the peer switches from channel B to channel C, it only needs to release its video feeds in swarm B and find new feeds in swarm C. At the same time, it continues its uploading in swarm A. This channel churn introduces no disruption to the distribution swarms of A, B and C. With these more stable distribution swarms, a peer's viewing experience and channel-switching delay can be dramatically improved.
- **Cross-Channel Multiplexing.** VUD enables cross-channel resource sharing. In particular, distribution swarms can be properly provisioned and adapted in response to the evolving channel popularity and achieve a cross-channel "multiplexing gain".

However, decoupling viewing from uploading requires more upload bandwidth overhead. To minimize this overhead, we also propose a substream-swarming enhancement. With substream swarming, a peer in a distribution swarm only downloads a small portion of the video stream, called a substream, and uploads the substream to multiple viewers. This way, peers in distribution swarms act as bandwidth amplifiers. We will show that VUD combined with substream-swarming dramatically improves viewing and channel-switching performance without requiring significant upload-bandwidth overhead.

We make the following contributions in this paper:

- To the best of our knowledge, this is the first P2P live streaming design that strictly decouples peer viewing from uploading. It addresses two fundamental problems that exist in current P2P streaming system designs: channel churn and channel-resource imbalance.
- We propose a set of peer assignment and bandwidth allocation algorithms to realize the proper provision of bandwidth among distribution groups. It eliminates the problem of resource shortage in small channels. When channel popularity changes, the algorithms can adjust bandwidth allocation dynamically.
- We introduce a substream enhancement to reduce the bandwidth overhead incurred by decoupling of viewing and uploading. We provide an analytical analysis of bandwidth overhead of VUD using substreams.
- We evaluate the performance of VUD via extensive simulations. The simulation results show that VUD significantly reduces switching delay, chunk miss ratio and playback lags. Moreover, VUD greatly improves the streaming quality of small channels. We also perform a preliminary PlanetLab evaluation of VUD.

The remainder of this paper is structured as follows. Section II summarizes related work. The detailed design of our cross-channel streaming system is presented in Section III. In Section III-A, we analyze the bandwidth overhead of our design. In Section IV, we describe the simulation methodologies and results related to the experiments conducted to verify the performance. In Section V, we present the experimental results obtained by PlanetLab evaluation. The paper is summarized in Section VI.

II. Related Work

P2P video streaming has attracted lots of research activities in recent years [10], [11], [12], [13]. Existing P2P video systems fall into two categories: tree-based [14], [15] and mesh-based [1], [16], [17]. Most of previous research work focuses on the design and improvement of isolated-channel P2P streaming systems, paying little attention to the optimization of multi-channel P2P streaming systems. In the multi-channel setting, there are few related published papers. In [18], Wu et al. proposed a server bandwidth provisioning algorithm to adjust

the supply of server bandwidth to different channels dynamically. Liao et al. [19] introduced inter-overlay cooperation in their system called AnySee to balance the resources among channels, and optimize streaming paths. In [20], Gan et al. proposed a reputation-based incentive mechanism to stimulate peers with spare bandwidth in resource-rich channels to help peers in resource-poor channels. Although the above works consider the balance of bandwidth resource among channels, they cannot solve the fundamental performance degradation incurred by channel churn. Our design differs in that, by strictly decoupling peer viewing and uploading, we address both channel churn problem and channel-resource imbalance problem simultaneously. With proper peer assignment and bandwidth provision, our system is immune to channel churn and realize cross-channel multiplexing. The idea of cross-swarm bandwidth sharing has been proposed for P2P file sharing applications, e.g [21]. In contrast, our VUD design addresses much more stringent delay and bandwidth requirements of P2P streaming applications.

III. View-Upload Decoupling with Substreams

We now describe more specifically VUD. Each video is divided into substreams (for example, to create K substreams for a video, the video source could assign every Kth constant-size chunk to a substream). As illustrated in Figure 1, the server divides the channel into K substreams. For each substream, there is a subset of peers, called a *substream distribution group*. Server only uploads each substream to one peer in each distribution group. Peers in the same distribution group upload and download the substream in P2P fashion. Finally, each viewer downloads all substreams from all distribution groups. Below we list some critical properties and observations

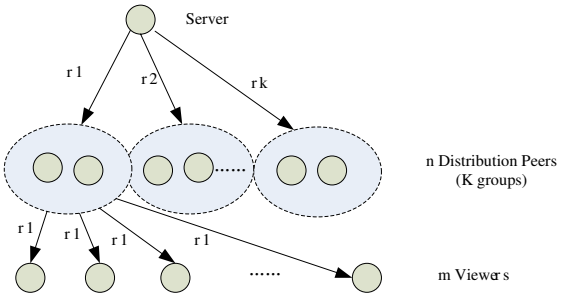


Fig. 1. Substream distribution groups in one channel: the server distributes one substream to each distribution group, which then distributes to the viewers.

about VUD:

1) The groups are semi-permanent, that is, the groups remain constant over medium time scales, and do not change as peers channel surf. However, the groups do evolve on a longer time scale to adapt to evolving channel popularity and peer churn.
2) If a peer is assigned to a distribution group for a substream, then it seeks to receive the complete substream. It redistributes the substream chunks to other peers in its distribution group and to peers outside the group that are currently viewing the corresponding channel.
3) A peer may belong to more than one distribution group, in which case it distributes more than one substream. If a peer is assigned to more than one subtream, it needs to allocate its upload bandwidth to its assigned substreams.
4) Intuitively, the aggregate upload capacity of a substream distribution group should reflect the demand for the substream. The greater the average channel demand, the larger the corresponding substream groups.
5) Intuitively, the peers in a distribution group should be chosen in regions that match the geographical demand. For example, if the majority of the demand for a channel is in Korea and the USA, then most of peers for the corresponding substream groups should be in Korea and the USA (and not, say, in Europe and China). However, in order to expose the intrinsic advantages of VUD, we postpone locality considerations to subsequent work.
6) Within a distribution group, we do not require the deployment of a specific distribution mechanism. We allow for trees [14], mesh-pull [1] and meshes with push-pull [17]. However, because the substream distribution groups are relatively stable, we should be able to employ tree-based mechanisms, which generally provide better delay performance than mesh-based mechanisms.

A. Limiting VUD Overhead

One immediate concern for the VUD design is its bandwidth overhead. The peers in a substream distribution group need to first download video of their substream, either directly from the server or from other distribution peers in the same group, before they can upload the substream to viewers. Consequently, the aggregate download demand for each substream increases proportionally with the number of distribution peers in that substream. We define the *VUD overhead* of a channel as the ratio between the total bandwidth (from the server and distribution peers) utilized to upload video to distribution peers and the required upload bandwidth to serve the viewers of this channel.

To study the VUD overhead, we focus on one VUD channel with streaming rate r, m viewers and n distribution peers. Distribution peer i has upload capacity of u_i. Immediately we have the following result:

Proposition 1: The VUD overhead of any channel with m viewers has an achievable lower bound of $\frac{1}{m}$.

Proof: For any distribution peer i, let d_i be the aggregate rate at which it downloads video from the server and other distribution peers, c_i be its upload contribution to all viewers. Peer i can maximize c_i by uploading downloaded video to all viewers in the channel. We have $c_i = \min\{u_i, md_i\}$, and VUD overhead can be calculated by

$$\xi \triangleq \frac{\sum_{i=1}^{n} d_i}{\sum_{i=1}^{n} c_i} \geq \frac{\sum_{i=1}^{n} d_i}{\sum_{i=1}^{n} md_i} = \frac{1}{m}.$$

The equality holds only if $md_i \leq u_i$, $\forall 1 \leq i \leq n$. One

scheme to achieve the lower bound is that the server uploads a substream of rate $\frac{u_i}{m}$ to distribution peer i, then distribution peer i uploads the substream to all m viewers at its full upload rate u_i. This way, all distribution peers act as a "bandwidth amplifiers" with a gain of m. ■

Essentially, to achieve the lowest VUD overhead, each channel is divided into many very fine substreams, each of which is distributed by a single distribution peer. However, for channels with a large number of viewers, it is impractical to have so many substreams. It is also unrealistic to have each distribution peer upload to all viewers. In practice, a channel is divided into a small number of substreams, each of which is distributed by a distribution group. The VUD overhead is determined by how substreams are divided and how distribution groups are formed. Next, we study how to design VUD to achieve a reasonably low overhead.

The VUD substream strategy will be characterized by the following variables for each substream:

- r_k: the substream rate for the kth substream group, with $\sum_{i=1}^{K} r_k = r$;
- $\mathcal{D}_k$: the group of distribution peers for substream k;
- $n_k = |\mathcal{D}_k|$: the number of distribution peers in $\mathcal{D}_k$, with $\sum_{i=1}^{K} n_k = n$;

Each distribution peer in $\mathcal{D}_k$ needs to download video at rate r_k. The total upload bandwidth consumed by all distribution peers in $\mathcal{D}_k$ is $n_k r_k$. Since the server only uploads one copy of substream k to one peer in $\mathcal{D}_k$, distribution peers will utilize $(n_k - 1)r_k$ of their upload bandwidth to disseminate the substream among themselves. To serve all viewers with substream k, distribution peers in $\mathcal{D}_k$ should upload to all viewers at an aggregate rate

$$mr_k \leq \sum_{i \in \mathcal{D}_k} u_i - (n_k - 1)r_k. \tag{2}$$

The channel VUD overhead can be calculated as

$$\xi = \frac{\sum_{k=1}^{K} n_k r_k}{\sum_{k=1}^{K} m r_k} = \frac{1}{mr} \sum_{k=1}^{K} n_k r_k$$

Proposition 2: When peers have homogeneous upload capacity, the substreams need to be divided equally to achieve the minimum VUD overhead.

Proof: When distribution peers have the same upload capacity $u_i = u$, the optimal VUD substream strategy to minimize the overhead can be obtained by solving:

$$\mathbf{argmin}_{\{r_k, n_k\}} \sum_{k=1}^{K} n_k r_k, \tag{3}$$

subject to:

$$mr_k \leq n_k u - (n_k - 1)r_k \tag{4}$$

$$\sum_{k=1}^{K} r_k = r \tag{5}$$

From (4), $n_k \geq \frac{(m-1)r_k}{u - r_k}$. Then we have

$$\sum_{k=1}^{K} n_k r_k \geq \sum_{k=1}^{K} \frac{(m-1)r_k^2}{u - r_k},$$

subject to (5). Since $f(x) \triangleq \frac{x^2}{u-x}$ is a convex function over $x \in [0, u)$, through Jensen's inequality, we have

$$\sum_{k=1}^{K} \frac{r_k^2}{u - r_k} \geq K * \frac{(\frac{1}{K}\sum_{k=1}^{K} r_k)^2}{u - \frac{1}{K}\sum_{k=1}^{K} r_k} = \frac{r^2}{uK - r},$$

with equality holds when $r_k = \frac{r}{K}$, $\forall k$. It can be easily verified that the minimum VUD overhead is

$$\xi^* = \frac{(m-1)r}{m(uK - r)},$$

which can be achieved when all substreams have the same rate and all distribution groups have the same number of peers $n_k = \frac{(m-1)r}{uK - r}$. ■

The minimum VUD overhead ξ^* decreases almost proportionally to peer uploading bandwidth and the number of substreams. For heterogeneous peers, the channel can also be divided into K equal-rate substreams. Distribution peers can be assigned to distribution groups to keep the average upload bandwidth within each group balanced, i.e., $\bar{u}^k \approx \bar{u}$, $\forall k$, where $\bar{u}^k$ is the average upload bandwidth in group k. Following the similar procedure as in the homogeneous case, it can be shown that the achieved VUD overhead is approximately $\frac{(m-1)r}{m(\bar{u}K - r)}$. If the system resource index is $\rho = 1.2$, the average distribution peer upload bandwidth is 1.2 times the streaming rate. It is sufficient to divide the channel into ten sub-streams to bring the VUD overhead down to 9%. VUD overhead will be *implicitly* bounded when we assign peers to achieve high resource index for all channels, we will discuss this in the following section.

B. Adaptive Peer Assignment

In this section, we present an adaptive algorithm that (i) assigns peers to substream groups; and (ii) for each peer, determines the fraction of upload capacity the peer assigns to each substream it is handling. Ideally, we would like such an assignment to balance the resource index across substreams and adapt to variations in channel demand. To this end, we introduce the following notation:

- K: the total number of substreams among all the channels;
- n: the total number of peers;
- r_k: the video rate of substream k;
- u_s: the upload capacity of servers;
- u_s^k: the server bandwidth allocated to substream k;
- u_i: the upload capacity of peer i;
- d_i^k: $d_i^k = 1$ iff peer i is in distribution group k;
- u_i^k: the allocated upload bandwidth of peer i for distributing substream k;
- u_{min}^k: the minimum upload bandwidth required to join distribution group k. (If $u_i^k \leq r_k$, peer i downloads more than uploads in group k. We require $u_i^k \geq 2r_k$);

- n_k is the number of peers in substream distribution group k, $n_k = \sum_{i=1}^{n} d_i^k$;
- m_k is the average number of peers viewing substream k over a short time scale (say, 5 minutes).

In assigning peers to groups, and allocating upload bandwidth to substreams, our primary goal is to balance the resource index across substreams. For a substream demand profile $\{m_k, k = 1, \ldots, K\}$, one can search for an optimal peer assignment $\{d_i^k\}$ and peer bandwidth allocation $\{u_i^k\}$ to balance the resource indexes $\{\rho_k\}$ across all substreams:

$$\max_{(\{d_i^k\},\{u_i^k\})} \min_k \rho_k = \frac{u_s^k + \sum_i u_i^k}{r_k(m_k + n_k)}, \tag{6}$$

subject to

$$d_i^k u_{min}^k \le u_i^k \le d_i^k u_i, \quad \sum_{k=1}^{K} u_i^k \le u_i, \tag{7}$$

which reflects that a distribution peer's upload bandwidth constraints. The problem is a NP-hard mixed fractional problem. We will instead resort heuristic algorithm to solve it. As the average demands $\{m_k\}$ evolve, and as peers churn (on a much larger time scale than channel churn), we will need to adapt the assignments and allocations accordingly. A secondary goal is that, when adapting to peer churn and long-term channel popularity evolution, we more often adapt by modifying the allocations rather than re-assigning peers to groups.

To this end, order the substream groups in ascending order according to their resource indexes ρ_k. Let G be the maximum number of groups that a peer can initially be assigned to. (In our simulations, we use $G = 5$.) When a new peer i with upload capacity u_i joins the system, we need to assign it to substream distribution groups, and then allocate its upload bandwidth to the assigned substreams. We consider assigning i to the first G substreams in the ordered list (that is, to the G most resource-poor substreams). To this end, we use a water-filling policy and distribute u_i among the G groups to balance the resource indexes of the G groups. In doing the water leveling, we always make sure that the allocated bandwidth u_i^k to any group is never less than u_{min}^k.

Because peers can leave the system and the average demand for viewing channels can change, we may also need to adapt assignments and allocations inbetween peer arrivals. In particular, if the resource index of a substream drops below a threshold ρ_{min}, we adjust the peer bandwidth allocations without (if possible) modifying the peer assignments to distribution groups. With fixed peer assignment, the optimization problem defined in (6) $\sim$ (7) is simplified into a linear max-min programming problem. Using the Lagrangian relaxation method [22], it can be solved by distributed algorithms implemented on individual peers. In our current experiments, all peering connections are TCP connections. The bandwidth allocation among all neighbors of a peer is regulated by the TCP congestion control scheme. We will investigate the optimal distributed bandwidth allocation algorithm in future work.

Algorithm 1: Peer Reassignment Algorithm

```
Initialize H={distribution groups with ρ ≤ ρ_min } ;
Initialize P = ∅ ;
while H ≠ ∅ do
    Select the distribution group k with the highest ρ;
    while ρ_k ≥ ρ_avg do
        Remove the peer with the lowest utilization of
        upload bandwidth from group k, and add the peer
        into P;
    end
    Sort the peers ∈ P in descending order according to
    their available upload bandwidth;
    for each peer v ∈ P do
        From the first to the last, assign v to a distribution
        group j in H that minimizes ρ'_j = (u_s^j + Σ u_i^j + u_v)/(m_j r_j);
        If ρ'_j > ρ_min, remove group j from H;
    end
end
```

After bandwidth allocation adjustment, if some distribution groups still have resource indexes below ρ_{min}, we will have to adjust peer assignments. The basic idea is to shift nodes from resource rich groups to the groups whose resource indexes are below the threshold. First, we select the distribution group k with the highest resource index, and continuously move the distribution peers with the lowest upload bandwidth utilization from that group to a set P until the resource index of group k falls below ρ_{avg}. Then, we sort all the peers in P into a descending-ordered list based on their available upload bandwidth. After sorting, for each peer v in P, from the first to the last, we assign v to a distribution group j with the minimum updated resource index $\rho'_j = \frac{u_s^j + \sum u_i^j + u_v}{m_j r_j}$. Let H be the set of distribution groups with $\rho \le \rho_{min}$. If the updated resource index $\rho'_j > \rho_{min}$, remove distribution group j from the set H. If there are still some resource-index poor distribution groups in H after all the peers in P have been assigned, we will choose the distribution group with the second highest resource index and continue the same process as above. The process will be executed until all the resource-index poor distribution groups have been removed from H. The details of peer reassignment algorithm is given in Algorithm 1.

IV. Simulation Experiments

A. Methodology

To evaluate the performance of VUD-based P2P streaming system, we implemented an event-driven P2P streaming simulator based on the source code provided by [23]. The simulator can simulate packet-level transmission and end-to-end latency among end nodes. It was further enhanced to support multiple channels and channel switching behaviors of peers.

In our simulation, as commonly assumed in P2P system study, we simulated an environment where the peer access links are the only bandwidth bottlenecks. In addition, we assume all

peers have enough bandwidth to download the channel they are watching and the substream they are distributing. To simulate the bandwidth heterogeneity among peers, we configure the upload capacity of peers according to Table I. The fraction of peers in different categories is calculated based on the measurement results in [24]. We only consider residential broadband connections, and Modem/ISDN connections are removed as they don't have enough download bandwidth. In the default

Type	Upload Capacity	Fraction of peers
1	768 Kbps	0.60
2	384 Kbps	0.25
3	128 Kbps	0.15

TABLE I
UPLOAD CAPACITY CONFIGURATION

simulation settings, there are totally 50 streaming channels and $2,000$ peers in the system. For each channel, there is only one server whose upload capacity is configured as 1 Mbps. For simplicity, all the channels have the same streaming rate of 400 Kbps. The video stream is further divided into 5 substreams for substream swarming.

Within each distribution group, the distribution peers are organized into a mesh and use *Push-Pull* method for chunk scheduling. Viewers of a channel also use Push-Pull method to fetch chunks from all substream distribution groups of the channel. By using push-pull method, the peers first use pull requests to get chunks from its neighbors. When it receives several continuous chunks from one neighbor, the peer can subscribe one or more substreams from that neighbor. In the following, the neighbor will relay the subscribed substreams directly to this peer without requiring any explicit request.

In our simulation, channel popularity follows Zipf-like distribution, as Zipf-like popularity distribution is reported in both P2P-based [4] and Telco-managed IPTV services [6]. Peers arrive at the system according to a Poisson process. In the default setting, the average time that peers stay in the multi-channel system is 4000 seconds. After joining in a channel for a period, the viewer starts to switch to another channel. The probability that one channel is chosen as the destination channel is proportional to its popularity. The peer continues to switch its viewing channel until it leaves the system. The time between two channel switchings is defined be *Channel Session*. In our simulation, the distribution of channel session follows a similar distribution as shown in [6].

For comparison, we also implemented a Push-Pull isolated-channel P2P streaming system similar to GridMedia [17], in which multiple channels are isolated from each other. Viewers of each channel form a separate mesh and use Push-Pull scheduling to exchange chunks. It serves as a baseline to evaluate the performance of VUD-based system. In the following, the baseline system is referred as *ISO*, and our VUD-based system is referred as *VUD*.

B. Performance Metrics

The following metrics are used for the comparison:

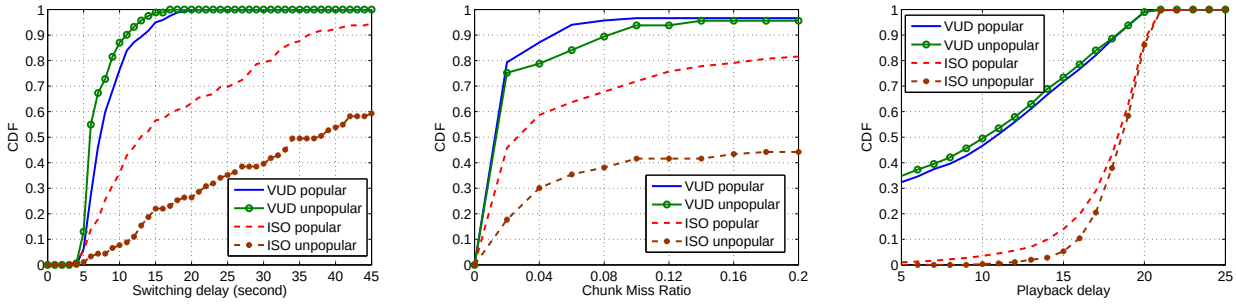
1) *Switching Delay:* Switching delay refers to the interval from one channel is selected until the initial buffer has been filled. In P2P video systems, the initial buffer is used to guarantee continuous playback and handle the rate variations of streaming sessions. It is always desirable to achieve shorter switching delay.
2) *Chunk Miss Ratio*: To enable smooth playback, chunks should arrive before its playback deadline. The miss chunks will cause video playback freezes for viewers and impact the viewing quality. Here, chunk miss ratio is defined as the number of chunks that miss the playback deadline over the total number of chunks that peer should receive. It is expected to minimize chunk miss ratio as much as possible.
3) *Playback Delay*: Playback delay refers to the interval from a chunk is generated at the source node to the moment it is played at the peer. In case that playback delay is large, the peer has to watch frames long behind the source.
4) *Control Message Overhead*: Peers exchange with their neighbors control information about their peer lists, chunk and substream availability, etc. Control message overhead is defined as ratio between the control traffic volume over the video traffic volume.
5) *VUD Bandwidth Overhead*: In VUD-based approach, the bandwidth allocated to distribution peers is extra overhead. In our experiment,*VUD bandwidth overhead* is defined as the ratio between the total upload bandwidth utilized to upload video to distribution peers and the total upload bandwidth used to upload video to viewing peers.

C. Simulation Results

In the experiment, peers start to switch their channels after the system enters steady state, and channel churn happens more frequently than peer churn. Each node maintains a streaming buffer of 35 seconds in terms of playback time, and starts playback only after it has filled the initial buffer (i.e., 100 continuous chunks). In the following part, we present the experimental results obtained from our simulation.

In Figure 2(a), we plot the distribution of switching delay under VUD and ISO design. We use "*popular*" (or "*unpopular*") to indicate the delay when switching to a popular (or an unpopular) channel [1]. From the figure, we observe that, for both popular and unpopular cases, VUD design spends much less time in accumulating enough continuous chunks and have much shorter channel switching delay (mostly less than 10 seconds) than ISO design. With ISO design, there exists significant difference in switching delays to popular channel and unpopular channel. The switching delay to a unpopular channel is much longer than to a popular channel. In contrast, the switching

[1]in the following, we use popular (or unpopular) similarly to indicate the performance in a popular (or unpopular) channel.



(a) CDF of Switching Delay (b) CDF of Chunk Miss Ratio (c) CDF of Playback Delay

Fig. 2. Performance comparison between ISO and VUD design through simulations: a) the switching delays to both popular and unpopular channels under VUD design are much smaller than those under ISO design. b) VUD design achieves much lower chunk miss ratio than ISO design for both popular and unpopular channels. c) VUD design has smaller playback delay compared with ISO design.

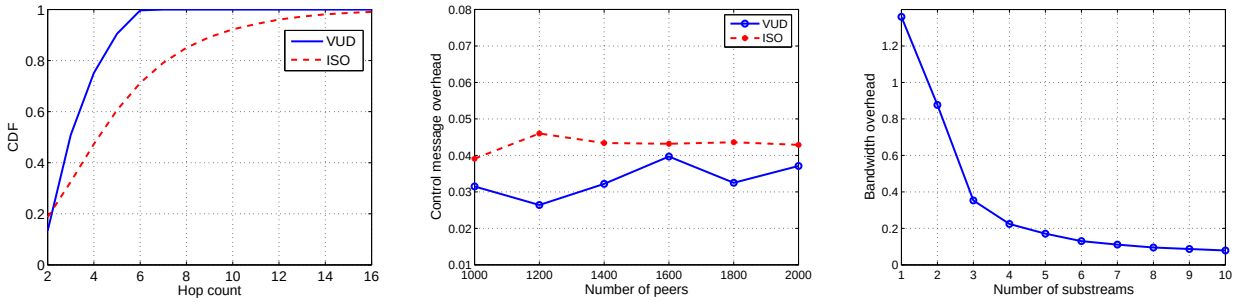
delay under VUD design is insensitive to channel popularity. The switching delays are comparably small for popular channel and unpopular channel. This is because the distribution peers in VUD design are quite stable, and the new joining peers can quickly find distribution peers that they can download chunks from. By decoupling viewing and distributing, VUD design efficiently solves the long startup delay problem under high channel churn.

Figure 2(b) compares the CDF of chunk miss ratio under two designs. Chunk miss ratio directly determines the playback continuity on peers. It is observed that, channel churn causes very high chunk miss ratio under ISO design. The situation becomes even worse for unpopular channels. In the ISO design, when a peer switches its viewing channel, it causes the same disturbance to its neighbors as if it leaves the system. Its neighbors have to spend time to find new peers to download data from and miss lots of chunks in the transient periods. One possible approach to reduce chunk miss ratio in ISO design is to adopt a larger buffer to mitigate "absorb" fluctuations from peer and channel churn. However, long buffers will prolong the switching and playback delays. Different from ISO design, VUD is designed to be immune to channel churn, as the viewers receive substreams directly from distribution peers. When a viewer switches its channel, it doesn't impact other viewing peers. From the figure, we can see that, around 80% peers in VUD design has a chunk miss ratio less than 0.02. As VUD design provisions bandwidth resource for each channel according to their demands, we also observe that there is not much difference for peers in different channels in terms of chunk miss ratio. For both popular and unpopular channels, the viewing peers have comparably low chunk miss ratios.

In Figure 2(c), we present the CDF of playback delay for both VUD and ISO designs. Under VUD design, the playback delays of about 30% viewers are less than 5 seconds, and that of 70% viewers are less than 15 seconds. In our current experiment, push-pull method is used for chunk scheduling among distribution peers. After the initial transient period, semi-static push relationship will be formed among distribution peers, and video chunks can be pushed quickly from the source to distribution peers, and finally relayed to viewing peers. Under ISO design, the playback delays are much longer for both popular and unpopular channels. The playback delays for about 90% of peers is longer than 15 seconds. As we use a small initial buffer (i.e., 100 chunks) in our simulation, most of playback delays under ISO design are less than 25 seconds. In case that peers use a large streaming buffer to accommodate channel/peer churn, the playback will be further increased. For VUD design, as the distribution peers are stable in medium time scales, the viewing peers can use a much smaller buffer, and achieve shorter playback delay.

To further understand the reduction of playback delay in VUD design, we also compare the hop counts of packets received by one viewing peer in a given channel in Figure 3(a). Packet hop count represents the number of hops that a packet traversed starting from the source node. It is observed that, most packets arrive at the viewing peer in less than 6 hops under VUD design, while the hop count under ISO design is much larger, and the hop count of about 20% packets is larger than 7. Figure 3(b) illustrates the control message overhead under different network sizes for VUD and ISO design. As VUD and ISO use similar push-pull method in chunk scheduling, their control message traffic is comparable and less than 5% of the total video volume. The control message overhead under VUD design is slightly lower than that under ISO design. The possible reason may be that, after stabilization, semi-static push relationship has been constructed among distribution peers, and fewer chunk request messages will be sent out.

In Figure 3(c), we measure the bandwidth overhead of VUD design. It is important for the feasibility and efficiency of VUD design in real systems. By varying the number of substreams, we can reduce the bandwidth overhead to a rather low level. For example, with 10 substreams, the bandwidth is even lower than 10%. Since VUD design can achieve much better performance than ISO design, such as quicker switch delay, smaller chunk miss ratio and smaller playback delay, the small bandwidth overhead can be well justified. To study the performance of VUD-based design under peer churn, we simulate the system by varying the average time that peers stay in the multi-channel



(a) CDF of Hop Count (b) Control Message Overhead (c) VUD Bandwidth Overhead

Fig. 3. Performance comparison between ISO and VUD design through simulations: a) chunks traverse fewer hops to reach peers, leading to shorter streaming delays. b) VUD design experience slightly lower control message overhead. c) the bandwidth overhead of VUD design decreases as the number of substreams increases.

system. When the average lifetime is shorter, the level of churn is higher. Figure 4 presents the average chunk miss ratio in both popular and unpopular channel under different peer churn levels. It is observed that, even when churn rate is high (e.g., expected lifetime = 0.2 hours), VUD-based design still can achieve low chunk miss ratio (less than 5%) by proper bandwidth adaptation and peer assignment.

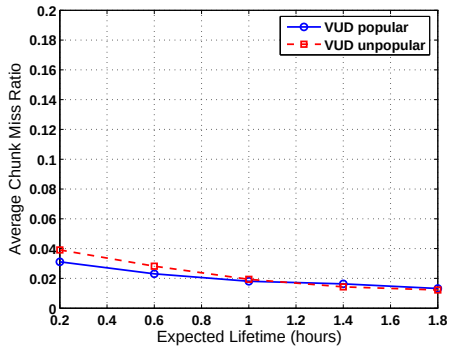


Fig. 4. Impact of Peer Churn on VUD: the average chunk miss ratio is still quite low when the level of churn increases.

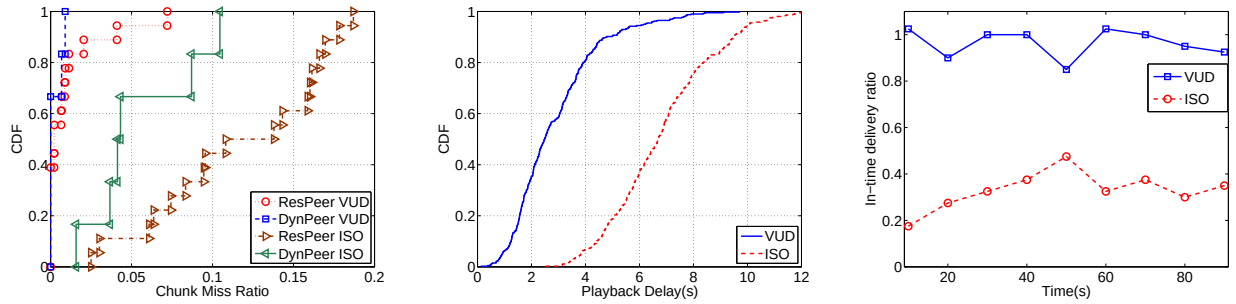
V. PlanetLab Experiments

To further demonstrate the advantage of VUD in real network environment, we also developed a prototype of VUD P2P streaming system and conducted an experiments on PlanetLab[25]. We set up two channels at video streaming rate of 512 kbps. Each channel has 24 viewers. The video source server also functions as tracker to provide peer list for new peers. Each viewer has homogeneous upload bandwidth of 600 kbps and source server of each channel has 1.28 Mbps bandwidth. For the isolated design, each channel is distributed in a mesh of 24 viewers and each peer is set to have 4 neighbors connected to exchange data. With VUD, each channel is divided into 8 substreams, and each substream is distributed by a swarm of 3 distribution peers, each of which can upload one substream to 8 peers. In both cases, peers have 10 second playback buffer window. To compare the performance of isolated design and VUD, we introduce a batch channel churn by swapping 6 peers between the two channels after all peers join the system and enter into stable state. We compare the chunk loss ratio and delay performance of two designs in the transient period of the first 100 seconds after the swapping.

Figure 5(a) shows the distribution of chunk missing ratio for peers switching channels (DynPeer) and peers staying in the same channel (ResPeer). In the isolated design, all the peers suffer high chunk miss ratio in the transient period. Peers in VUD experience very few chunk misses. Most of them have less than 5% chunk miss ratio. The chunk generation time at the source side and the arrival time at peer side have been recorded during the experiment to calculate the delay for each chunk. The distribution of playback delays of selected channel switching peer are compared in Figure 5(b). The average chunk delay in VUD is less than 4 seconds, while the average delay for peers in ISO design is 7 seconds. Peers in VUD experience shorter chunk delays, which implies that the VUD distribution is more effective and users experience less chunk loss given playback deadline. Chunks are disposed if they are received after their playback deadline. The ratio of chunks delivered within 6 seconds is plotted in Figure 5(c). VUD delivers most chunks in time, while the isolated design will miss the majority of chunks if the playback delay is set to 6 seconds. In more realistic systems, for which channel churn is more frequent, we expect VUD to have even larger performance improvement.

VI. Conclusion and Future Work

In this paper, we presented VUD, a novel streaming framework for multi-channel P2P video systems. By radically decoupling peer video uploading from viewing, VUD solves two fundamental performance problems of the traditional isolated channel P2P streaming, namely, excessively long channel switching delays and poor quality for small channels. We demonstrated through simulations and experiments on PlanetLab that VUD is immune to high channel churn and can efficiently achieve the multiplexing gain between channels with diverse popularity. In addition, we showed analytically and experimentally that VUD overhead can be well managed through balanced substreaming and peer assignment.



(a) (b) (c)

Fig. 5. Performance comparison between ISO and VUD design in PlanetLab experiments: a) chunk miss ratios on channel-switching peers and stable peers. b) chunk delay distribution on a selected channel switching peer. c) ratio of chunks delivered within 6 seconds.

Our study opens up a new design space for the future multi-channel P2P video systems. While our initial results are encouraging, lots of immediate future work needs to be done to fully explore the advantages of the VUD design. In Section III, we proposed simple heuristics for peer assignment and bandwidth allocation. We will enhance the algorithms to better adapt to dynamic channel popularity and, more challengingly, flash crowds. We will also consider heuristics that take into account ISP locality to largely reduce the traffic imposed on ISP networks. Although distribution swarms can be managed by a centralized tracker, to address the scalability of systems with many channels, we will study DHT-based distributed solutions for cross-channel resource allocation. In addition, we will also consider provisioning peers to distribution groups to match the geographical demand.

The VUD framework is open to any P2P streaming design. In the experiments, distribution swarms inherit a generic mesh-based design for isolated streaming systems. By exploiting the largely improved stability of distribution swarms in VUD, we will develop new substream-based swarm structures and scheduling algorithms that balance the needs of performance, adaptivity and robustness. In our simulation, the substreams are simply obtained by partitioning each video with time-division multiplexing. A peer needs to receive all substreams in the channel to achieve acceptable viewing quality. We will investigate using layered video [26], [27] to create the substreams. With layered video, a peer with even only one substream can start video playback. Peers receiving different subsets of substreams perceive different video quality. New VUD substream swarming will be designed to achieve video quality differentiations among heterogeneous peers.

REFERENCES

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

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

[3] "Sopcast Homepage," http://www.sopcast.com/.

[4] 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*, December 2007.

[5] X. Hei, Y. Liu, and K. Ross, "Inferring Network-Wide Quality in P2P Live Streaming Systems," *IEEE Journal on Selected Areas in Communications*, December 2007.

[6] M. Cha, P. Rodriguez, S. Moon, and J. Crowcroft, "On Next-Generation Telco-Managed P2P TV Architectures," in *Proc. of IPTPS*, 2008.

[7] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *Proc. of ACM SIGCOMM*, 2001.

[8] R. Kumar, Y. Liu, and K. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. of IEEE INFOCOM*, 2007.

[9] C. Liang, Y. Guo, and Y. Liu, "Is Random Scheduling Sufficient in P2P Video Streaming?" in *Proc. of ICDCS*, June 2008.

[10] P. Francis, "Yoid: Extending the Internet Multicast Architecture," Cornell University, Tech. Rep., April 2000.

[11] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. of ACM SIGCOMM*, October 2002.

[12] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *Proc. of ACM SIGMETRICS*, 2000.

[13] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2cast: Peer-to-peer patching scheme for vod service," in *Proc. of WWW*, May 2003.

[14] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," in *Proc. of ACM SOSP*, 2003.

[15] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *Proc. of ACM SOSP*, 2003.

[16] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *Proc. of IPTPS*, 2005.

[17] M. Zhang, L. Zhao, Y. Tang, J.-G. Luo, and S. Yang, "A peer-to-peer network for streaming multicast through the internet," in *Proc. of ACM Multimedia*, 2005.

[18] C. Wu, B. Li, and S. Zhao, "Multi-channel Live P2P Streaming: Refocusing on Servers," in *IEEE INFOCOM 2008*, Apr. 2008.

[19] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-Peer Live Streaming," in *IEEE INFOCOM 06*, 2006.

[20] G. Tan and S. Jarvis, "Inter-Overlay Cooperation in High-Bandwidth Overlay Multicast," in *ICPP-06*, Aug. 2006.

[21] L. Guo, S. Chen, Z. Xiao, E. Tan, and X. D. X. Zhang, "Measurements, analysis, and modeling of bittorrent-like systems," in *Internet Measurement Conference (IMC 05)*, Berkeley, California, USA, Oct. 2005.

[22] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[23] M. Zhang, "Peer-to-Peer Streaming Simulator," http://media.cs.tsinghua.edu.cn/~zhangm/download/.

[24] C. Huang, J. Li, and K. Ross, "Can Internet VoD be Profitable?" in *Proc. of ACM SIGCOMM*, 2007.

[25] "PlanetLab Homepage," http://www.planet-lab.org.

[26] M. Wien, H. Schwarz, and T. Oelbaum, "Performance analysis of SVC," in *IEEE TCSVT*, vol. 17, no. 9, September 2007.

[27] M. Wien, R. Cazoulat, A. Graffunder, A. Hutter, and P. Amon, "Real-time system for adaptive video streaming based on SVC," in *IEEE TCSVT*, vol. 17, no. 9, September 2007.