Enabling Broadcast of User-Generated Live Video without Servers

ABSTRACT

We are witnessing the unprecedented popularity of User-Generated-Content (UGC) on the Internet. While YouTube hosts pre-recorded video clips, in near future, we expect to see the emergence of *User-Generated Live Video*, for which any user can create its own temporary live video channel from a webcam or a hand-held wireless device. Hosting a large number of UG live channels on commercial servers can be very expensive. Server-based solutions also involve various economic, copyright and content control issues between users and the companies hosting their content. In this paper, leveraging on the recent successes of P2P video streaming, we study the strategies for end users to directly broadcast their own live channels to a large number of audiences without resorting to any server support. The key challenge is that end users are normally bandwidth constrained and can barely send out one complete video stream to the rest of the world. Existing P2P streaming solutions cannot maintain a high level of user Quality-of-Experience (QoE) with such a highly constrained video source. We propose two strategies to address this challenge. We first propose a proactive-push based source-side scheduling algorithm to increase the scale of P2P broadcasts that can be driven by end users. We then propose a novel layered P2P streaming architecture that introduces peer playback delay differentiations to further boost end users' capability of driving large-scale video streaming. Through detailed packet-level simulations and PlanetLab experiments, we show that the proposed strategies enable a source with upload bandwidth slightly higher than the video streaming rate to stream video to tens of thousands of peers with premium quality of experience.

1. INTRODUCTION

User-Generated-Content (UGC) has become tremendously popular on the Internet in recently years. The global connectivity provided by the Internet makes it extremely easy for users to share a wide variety of UGC, including blogs, photos and video clips. YouTube [1], the popular UGC video streaming site, serves 100 million distinct videos and attracts $65,000$ uploads daily. While Youtube offers pre-recorded video, in the upcoming years, we expect to see the emergence of *User-Generated Live Video*, for which any user can create its own temporary live video channel from a webcam or a hand-held wireless device. The live channel could be a professor's lecture, a little-league baseball game, a wedding, an artistic performance, or a political demonstration. Unlike pre-recorded video, live video streaming has to meet the stringent video playback deadlines. The dissemination of user-generated live video to a large number of audience is a challenging problem, and is the focus of this paper.

One may naturally resort to a server-based solution and build a live version of YouTube. Server farms and CDN networks can be employed to host UG live channels and stream videos directly to viewers. However, the server and network infrastructure cost grows proportionally to the number of viewers. YouTube is estimated to pay over 1 million dollars per month to content distribution network (CDN) for distributing videos [2]. Similarly, it will be very expensive to host a large number of UG live channels. Peer-to-Peer (P2P) technology has recently been adopted to offload servers by efficiently utilizing the upload bandwidth of end users. Dozens of commercial P2P video streaming systems, such as PPLive [3] and PPStream [4], have been deployed on the Internet to deliver live and on-demand video services. Although existing P2P solutions can offload server efficiently, a reasonably large server bandwidth is still assumed to achieve good streaming performance, such as low startup delay and chunk loss ratio, in large scale P2P streaming. Current commercial P2P IPTV systems still invest considerably on servers that track peers, host video and bootstrap P2P video sharing. Technically, it is possible for those companies to provide a service to host UG live channels in the near future. However users and companies will have to reach agreement on various economical, legal, copyright and content control issues to make it happen.

In this paper, we investigate a "pure" P2P streaming solution that enables a user to broadcast his/her UG live video to a large number of audiences without any server support. In our solution, any user generating a live video can act as a video source and directly drive a P2P live broadcast to users interested in his/her video. Unlike a commercial video streaming server, the user generating the video only has an upload bandwidth slightly higher than the encoded video rate and can barely send out one complete video stream to the rest of the world. The challenge is how to drive a small, medium or even large scale P2P live streaming using such a constrained source. We propose two strategies to address this challenge. First we investigate the impact of source-side chunk scheduling on the system streaming performance. We show that, by simply switching the source-side chunk scheduling from *passive* to *proactive* mode, a source with upload bandwidth slightly higher than the streaming rate can drive a small or medium scale P2P live broadcast. In P2P streaming, a certain amount of video playback delay on peers is necessary to facilitate video sharing. The playback delays tolerated by peers in current commercial P2P IPTV systems are on the order of tens of seconds [5]. Leveraging on peer's playback tolerance, we propose a Layered P2P Streaming (LPS) architecture that introduces peer playback delay differentiations and constructs virtual servers out of peers to drive large-scale streaming. In LPS, a small fraction (e.g., 5%) of peers are assigned to an *amplifier* layer, and the rest of peers are assigned to a *base* layer. Peers at the amplifier layer have shorter target playback delays than peers at the base layer. The source only serves a small number of amplifier layer peers, which will then forward downloaded video to peers at the base layer. Effectively, there are multiple virtual "servers" from the amplifier layer that collaboratively deliver a high level of video Quality-of-Experience to a large number of peers at the base layer.

The contribution of this paper is three-fold:

1. We study the impact of source-side chunk scheduling and propose a proactive push-based algorithm to increase of the scale of P2P broadcast that can be driven by a bandwidth-constrained source.

2. We dissect various delay components in P2P streaming. By differentiating peer playback delays, we propose a layered P2P streaming architecture to further boost source's capacity in driving large-scale P2P streaming. We develop a robust mesh-based implementation for the proposed LPS architecture. A set of peer assignment and management algorithms are also developed. To the best of our knowledge, our work is the first one to explicitly differentiate peer playback delays to improve the performance of P2P streaming.

3. To evaluate the performance the proposed solutions, we build a detailed packet-level P2P simulator and conduct extensive simulations driven by traces from real systems. Our simulation results demonstrate that the proposed strategies enable bandwidth-constrained source to stream video to a large number of peers with excellent video quality and low startup delay. We also develop a prototype and conduct experiments on PlanetLab to examine the feasibility and performance of LPS in real Internet environment.

The remainder of this paper is organized as follows. We briefly discuss the related work in Section 2. The strategy of source-side chunk scheduling is discussed in Section 3. The LPS architecture and detailed system implementation is presented in Section 4. The simulation setting and numerical results are presented in Section 5. The paper is concluded in Section 6.

2. RELATED WORK

P2P technology has made great progress since its debut, and been widely applied to various content distribution applications, such as file sharing and video streaming.Traditionally, P2P live streaming systems can be broadly classified into two categories, namely tree-based and mesh-based. In tree-based approach (such as ESM [6]), peers form multiple multicast trees in application level and relay traffic as interior nodes. The multiple trees are optimized further in terms of peer bandwidth, peer load and latency in SplitStream [7] and Chunkyspread [8]. Compared with tree-based approach, the mesh-based ones have been widely adopted by current commercial systems due to their lightweight management requirement and robustness to peer churn. Peers form a overlay network and dynamically exchange data with their neighbors. Many mesh-based P2P streaming systems have been proposed, such as Chainsaw [9], DONet/CoolStreaming [10] and PRIME [11]. Hybrid streaming solutions combine tree-push and mesh-pull schemes. The work in [12] tries to improve video chunk scheduling efficiency with sub-stream based push method. Peers establish the parent-child relationships and sub-streams are pushed from parents to their children. mTreebone [13] constructs a single tree with stable nodes rooted at the server and delivers video via a combination of pushing through the tree and pulling through the underlying auxiliary mesh. To assess the stability of nodes, it needs to know the **lifetime of the channel beforehand ??**. Frequent tree adaptation is also needed to optimize the tree performance. Besides the overlay topology construction and scheduling, other aspects of P2P streaming have also been investigated in a rich literature [14, 15, 16, 17, 18].

However, the previous live streaming systems generally endeavor to optimize the performance under the assumption that the server has reasonably large bandwidth to drive the system. From the perspective of fluid theory, the authors in [19] have analyzed the supportable streaming rate based on server capacity and peer upload capabilities. The online server capacity provisioning algorithm proposed in *Ration* [20], dynamically adjusts the server bandwidth distribution among concurrent channels according to forecasted demands. The strategies with very constrained server bandwidth have not been explored so far. To support UG live channels, our work focuses on the scheduling and system architecture design to enable source with constrained bandwidth to stream video to a large number of peers. The proposed LPS architecture is the first one to deliberately differentiate the playback delays of peers in the same channel. We show that peer playback delay differentiation can "amplify" source upload capacity and lead to largely improved QoE in large-scale video streaming.

3. IMPACT OF SOURCE-SIDE CHUNK SCHEDULING

We first investigate the impact of source-side chunk scheduling on video streaming quality in the whole system. In traditional receiver-driven pull-based systems, a specialized server with reasonably large bandwidth receives new chunks from video source and operates as the delivery source of the P2P streaming overlay. The server broadcasts buffer-map periodically to directly connected peers. After the server receives the pull requests from peers, it replies with the requested chunks. The server acts passively in the content distribution process. Generally this works fine when the server has reasonably large upload bandwidth. New chunks can be sent out by the server in time and eventually be received by all peers. However, a source with bandwidth slightly higher than the streaming rate can be easily overwhelmed by peer requests for old chunks and cannot send the fresh chunks out. Undoubtedly, this will reduce the chunk diversity in the network and peer bandwidth utilization, and finally slow down the whole distribution process.

For a source with constrained bandwidth, passively waiting for pull requests is no longer sufficient to send out new chunks in time. It should be more proactive and push out new chunks to peers as soon as possible. A proactive source pushes the newly generated chunks to neighbors without going through the request-reply process. It is unnecessary for it to broadcast buffer-maps as well. This broadcast-once-generate procedure not only minimizes the time that a new chunk is queued on the source side, but also avoids blocking of new chunk transmissions by pull requests for older chunks. Hence it increases the content diversity in the system as well. In P2P streaming system, peers come and go frequently. Suppose the source s's upload bandwidth equals to the streaming rate, i.e., the source is only capable of sending one copy per chunk to its neighbor set $\mathcal{N}(s)$. Once one peer receives one fresh chunk from the source and leaves before it has a chance to pass it to other peers, this chunk loses the opportunity to be delivered to other peers in the system. We call this phenomenon *delivery loss*. To avoid delivery loss, the source needs to push complementary copies to peers. Therefore for a proactive source, its upload bandwidth should be at least slightly higher than the streaming rate for complementary push.

Algorithm 1: Chunk Scheduling of Proactive Source

```
   input : s($\mathcal{W}$,$\mathcal{N}$)
1  $c \leftarrow$ GenNewChunk()
2  $\mathcal{W} \leftarrow$ UpdateSlideWin($\mathcal{W}, c$)
3  PushToNbr($c, \mathcal{N}$)
4  while $bw_{avail} \neq 0$ do
5      $i \leftarrow$ FindRarestChunk($\mathcal{W}$,$\mathcal{N}$)
6      PushToNbr($i, \mathcal{N}$)
7  end
```

Algorithm 1 shows the corresponding push-based scheduling algorithm for the proactive source. Once the source generates a new chunk, it pushes to one peer from the neighbor set $\mathcal{N}$ according some policy, such as preferring the peer with large bandwidth in heterogeneous network. The sliding window $\mathcal{W}$ of the source will be updated to incorporate the new chunk, and also wipe off the oldest chunk. Meanwhile, source monitors the chunk availability on its direct neighbors by examining the received buffer-map information. Whenever the source has spare bandwidth, it proactively pushes out complementary copies of the rarest chunks to its neighbors. Generally, the newly generated chunk c would be the rarest one in case no delivery loss happens. This complementary source push mechanism can minimize the possibility of delivery loss without blocking deliveries of fresh chunks. For source with reasonably large bandwidth, the delivery loss rarely happens and the corresponding algorithm can be simplified further. The source just tries to push as many copies as possible in case a new chunk is generated. Otherwise it needs more protocol messages to prevent peers from receiving duplicate copies of old but rarest chunks since source pushes data without coordination with peers. Eventually, a chunk can be distributed to all peers successfully if no delivery loss happens to it.

To compare the performance of passive source and proactive source scheduling, we conducted detailed packet-level simulations. As will be shown in Section 5, the proactive push-based scheme greatly improves the system performance and enables a bandwidth constrained source to drive medium size network with satisfactory performance. Even with only 10 second buffer window size, the average delivery ratio of all $2,000$ peers can be kept above 97% by the source with bandwidth slightly higher than the streaming rate. However, as scale of the system continues to grow, the system performance with proactive source drops inevitably. The average delivery ratio drops to around 90% with the same setting when the population is more than $10,000$. To support a larger number of peers with premium quality of experience, we propose the layered P2P streaming architecture and discuss it in the following section.

4. LAYERED P2P STREAMING ARCHITECTURE

In this section, we introduce the layered streaming architecture for bandwidth constrained source to drive large-scale P2P video streaming. We further present a mesh-based implementation for this architecture.

4.1 Taxonomy of P2P Streaming Delay

First we dissect different delay components of P2P streaming systems. Fig. 1 presents the instantaneous buffer status of a peer

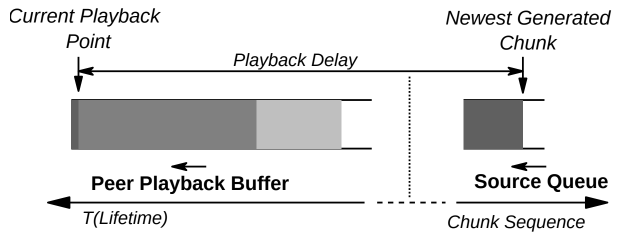


Figure 1: Delay Dissection in P2P Streaming System

and a source. The newest chunk with the highest sequence number is distributed into the P2P streaming overlay network by the source. The *lifetime* of a chunk is defined as how long it has existed in the system since it was generated at the source. At the peer side, the oldest chunk with the smallest sequence number is played and then disposed. Three different delay definitions in P2P streaming are listed as follows:

- *Playback delay* (d_{pb}) represents the lag of a peer's viewing progress from the source. If at time t, a peer is playing a video chunk generated by the source at time $t' < t$, the playback delay is $t - t'$.
- *Start-up delay* (d_{st}) represents how fast the video playback starts on user side after the application is launched. If a user launches the streaming application at time t, the playback starts at time $\bar{t} > t$, then the start-up delay is $\bar{t} - t$. Normally, the start of playback is triggered when the downloaded content in video streaming buffer reaches certain threshold during the start-up phase.
- *Chunk delay* (d_{cp}) measures how fast a chunk can be distributed to a peer after it is delivered to the network. If a chunk is delivered to the network at time t and a peer receives the chunk at time $\hat{t} > t$, the corresponding chunk delay to this peer is $\hat{t} - t$.

Playback delay indicates how closely the user viewing process is synchronized with the source progress. It is important for the broadcasting of live events, such as real-time news, sports events. Generally in current P2P IPTV system, the user playback delay tolerance may be more than ten seconds. Instead, users are more concerned with the following two performance metrics that influence their video experiences directly: 1) start-up delay. Everyone expects to get fast response from the application instead of waiting for a long time buffering. 2) playback quality. All chunks should be received before their corresponding playback deadlines to ensure smooth and continuous playback. Our layered P2P streaming architecture introduces small peer playback delay differentiations in order to shorten video start-up delays and chunk delays.

4.2 Architecture Overview

In LPS, peers are assigned to two different layers: the *amplifier layer* and the *base layer*. The majority of the peers are assigned to the base layer and the amplifier layer only consists of a very small fraction of strong peers. Peers at the same layer have the same target playback delay. The target playback delay d^a_{pb} of the amplifier layer is shorter than that of the base layer d^b_{pb}. The delay difference $\delta = d^b_{pb} - d^a_{pb} > 0$ is kept small. In LPS, peers within the same layer form a streaming overlay and exchange data chunks. The source only needs to drive a small scale P2P video streaming at the amplifier layer. Peers at the amplifier layer download and playback the new data content first. Then they become the potential seeds, named as *suppliers*, to serve the peers at the base layer. As a result, the large scale P2P streaming at the base layer is driven by multiple seeds in the "server cluster" at the amplifier layer. Effectively, peers at the amplifier layer amplify the bandwidth constrained source's capability in driving large-scale P2P streaming.

In this way, the peers at both layers experience largely shortened start-up delays and chunk delays. At the amplifier layer, the source now only serves a much smaller P2P streaming overlay network compared with the single flat large overlay network with the traditional approach. At the base layer, P2P streaming is driven by multiple seeds from the upper layer "server cluster". This design brings multiple advantages. Firstly, it eliminates the source bandwidth bottleneck by leveraging multiple seeds from upper layer whose aggregate capacity is far more than that of the bandwidth constrained source. Secondly, peers can choose a source nearby to download data from. It will reduce the number of hops that chunks have to traverse. Furthermore, although the playback progresses of the base layer peers are delayed by δ, with much improved streaming efficiency, we will show that a relatively short δ can achieve the most gain of LPS. This makes LPS an appealing solution for broadcasting of live events with moderate real-time requirement. Fig. 2 shows an simple example of the architecture. A bandwidth

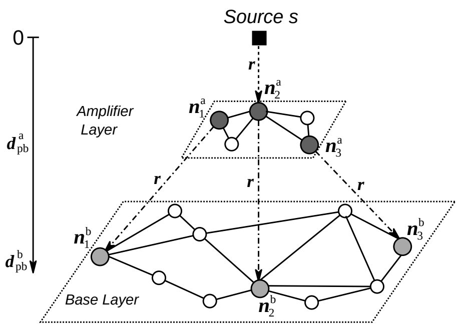


Figure 2: LPS Architecture

constrained source distributes nearly one copy of the video stream to the peers at amplifier layer. Peers n_i^a at the amplifier layer have abundant bandwidth and distribute one copy of data received to peers n_i^b at the base layer. The n_i^b peers then broadcast chunks to other peers at the base layer. In this case, the peers at the base layer receive three copies of data at streaming rate $3r$, and the source bandwidth amplified ratio reaches $\alpha_{amp} = 3$.

Multiple peers at the amplifier layer act as local video proxy for peers at the base layer. This largely reduces the chunk delays and improves distribution efficiency at the base layer. The collection of edges used for the delivery of a single chunk from the source to all participating peers form a source-rooted *delivery tree* [21] in either mesh-based or tree-based approaches. Suppose the source and all peers have homogeneous bandwidth of $u = 2r$, each node can have two children in the chunk delivery tree. Next we analyze the impact of amplified source bandwidth on chunk delay in single chunk delivery tree. With single source, there are 2^i peers at the ith level and $n_x = \sum_{i=1}^{x} 2^i$ peers for the tree with x levels in total. Given total N peers, we have $n_x = N$ and the distance from leaf peers to the root is $x_{max} = \log_2(N/2+1)$ hops. When $N = 2000$, $x_{max} \approx 10$. The majority of peers are located at the bottom level of the tree. We have $n_6/n_{10} = 6.2\%$, i.e., only 6.2% peers receive a chunk after 6 hop transmissions from the root. In LPS, each supplier at the amplifier layer can be a root for a delivery tree at the base layer. If there are α suppliers, (corresponding to the source bandwidth amplified ratio of α), there are $2^i\alpha$ peers at the i-th level of all delivery trees. Correspondingly the number of peers up to x levels is $n_x^{lps} = \sum_{i=1}^{x} 2^i\alpha$. The maximum level to cover N peers is $x_{max}^{lps} = \log_2(N/2\alpha+1) \approx x_{max} - \log_2\alpha$. We have $x_{max}^{lps} \approx 4$ when $\alpha = 64$ (In later simulation, α reaches 70 when $N = 2000$). All peers can receive the data in less than 4 hops in LPS, less than half of the previous one. In a nutshell, the amplified source bandwidth could tremendously increase the fan-out of data delivery trees and significantly reduce chunk delays.

4.3 Peer Assignment among Layers

In LPS, peers need to be properly assigned to the two layers to balance the bandwidth availability. For the amplifier layer, peers should have sufficient upload bandwidth to exchange video data among themselves. In addition, they should have extra bandwidth to upload video to peers at the base layer. Suppose there are M types of peers with different upload capacities $\{u_i, 1 \le i \le M\}$. Let p_i be the fraction type i peers, the average bandwidth is $\bar{u} = \sum_{i=1}^{M} p_i u_i$. For total N peers, the target sizes of the amplifier layer and base layer are N_a and N_b respectively, with $N_b = \beta N_a$, where β denotes the normalized load of the amplifier layer and $\beta >> 1$. Given a streaming rate r and source upload bandwidth u_s, the *resource index* for the entire system is defined as the ratio of the aggregate resource over aggregate demand.

$$\rho = \frac{u_s + N\bar{u}}{Nr} \approx \bar{u}/r \tag{1}$$

To let all peers receive the full video, ρ should be larger than 1. Let $\mathcal{S} \subset M$ be the subset of types of peers that can be the potential suppliers for the base layer peers and $u_i > r, i \in \mathcal{S}$. For a ith type supplier, it would contribute λ_i copies of streaming rate r to the base layer. And we define p_i^a as the fraction of type i peers at the amplifier layer. To achieve a source bandwidth amplify ratio larger than α^0, we should have

$$\alpha_{amp} = \sum_{i \in \mathcal{S}} N_a p_i^a \lambda_i r / u_s > \alpha^0. \tag{2}$$

To assure peers at the amplifier layer also achieve good performance, we assume the network size of amplifier layer should be less than N_a^0 and the resource index larger than ρ_a^0. Correspondingly we obtain

$$N_a = \frac{1}{\beta+1} N < N_a^0 \tag{3}$$

$$\rho_a = \frac{u_s + \sum_{i=1}^{M} N_a p_i^a u_i - \sum_{i \in \mathcal{S}} N_a p_i^a \lambda_i r}{N_a r} > \rho_a^0 \tag{4}$$

Peers should be assigned to amplifier layer to satisfy equations (2), (3) and (4).

To simplify the analysis, we choose only some kth type of peers to be the potential suppliers for the base layer peers, and $u_k > r, k \in \mathcal{S}$. Every supplier would contribute λ copies of streaming rate r to the base layer. We adjust the kth type peer distribution at the amplifier layer to p_k^a, and then the resource indexes of the two layers can be listed as follows.

$$\rho_a = \frac{u_s + N_a p_k^a (u_k - \lambda r) + N_a(1-p_k^a)\bar{u}'}{N_a r} \tag{5}$$

Equation (5) shows the resource index of the amplifier layer, where $\bar{u}'$ denotes the average bandwidth of peers at amplifier layer excluding the kth type, $\bar{u}' = \sum_{i \neq k} p_i^a u_i / \sum_{i \neq k} p_i^a$. Correspondingly, we have the resource index of the base layer as follows:

$$\rho_b = \frac{N\bar{u} - N_a p_k^a (u_k - \lambda r) - N_a(1-p_k^a)\bar{u}'}{N_b r} \tag{6}$$

Next we discuss the impact of the distribution of the supplier on the resource indexes of two layers. Equation (5)(6) show that the resource indexes of two layers change linearly with p_k^a, the fraction of k-th type peers at the amplifier layer. There are two extreme cases.

Case 1: There is no differentiation among peer distribution of two layers, then we have $p_i^a = p_i$ for ith type peer. In this case, peers could be randomly assigned to these two layers. The resource indexes of the two layers are

$$\rho_a^1 = \frac{u_s + N_a\bar{u} - N_a p_k \lambda r}{N_a r} \approx \rho - p_k\lambda$$

$$\rho_b^1 = \frac{N_b\bar{u} + N_a p_k \lambda r}{N_b r} = \rho + p_k\lambda/\beta$$

Case 2: The amplifier layer is formed only by the supplier type of peers, then we have $p_k^a = 1$. And $Np_k^a > N_a$ assures that there are enough number of supplier type of peer for the amplifier layer. Then we have the resource indexes

$$\rho_a^2 = \frac{u_s + N_a u_k - N_a\lambda r}{N_a r} \approx u_k/r - \lambda$$

$$\rho_b^2 = \frac{N\bar{u} - N_a u_k + N_a\lambda r}{N_b r} = \frac{\beta+1}{\beta}\rho - \frac{u_k/r - \lambda}{\beta}$$

When $\beta >> 1$, the resource index of the base layer at both cases ρ_b^1 and ρ_b^2 can be approximated by ρ. We only need to take care of the resource index of the amplifier layer. Based on Equation (5), we know that if the following condition holds, then the ρ_a will increase as the p_k^a increases.

$$u_k > \lambda r + \bar{u}' \tag{7}$$

The range of resource index of the amplifier layer is $[\rho - p_k\lambda, u_k/r - \lambda]$ as we increase the fraction of the kth type of peers. The above analysis gives guidelines on setting the appropriate value of parameters. Suppose $\beta = 20$ and $u_s = r = 400$kbps, we pick the 1Mbps

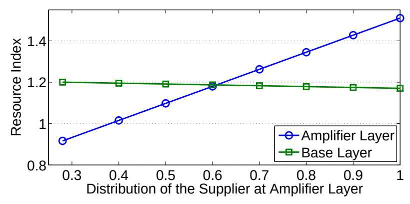


Figure 3: Resource Indexes of Two Layers

bandwidth peers as suppliers with original distribution $p_k = 28\%$ and $\lambda = 1$. The average bandwidth of $\bar{u}$ and $\bar{u}'$ are 475kbps and 270kbps respectively (same as later simulation setting). Fig. 3 shows the resource indexes of these two layers. The resource index of the base layer almost remains unchanged, while that of the amplifier layer can be adjusted as the p_k^a changes.

As we know from Equation (2), the source bandwidth amplified ratio will be increased if we increase the size of the amplifier layer N_a. However, since the source has constrained bandwidth and can only drive P2P streaming with limited scale at the amplifier layer, N_a should be kept small. In addition, one should always try to assign peers with larger bandwidth to the amplifier layer to increase the number of copies contributed to the base layer.

4.4 Implementation with Mesh-Pull based Approach

Various scheduling designs can be applied at each layer of LPS. We use mesh-pull based design as an example to discuss the implementation of LPS.

4.4.1 Lightweight Tracker-based Peer Assignment

Like regular mesh-pull based system, there is a centralized tracker that is responsible for the architecture construction and maintenance. When a new peer joins the system, it needs to register at the tracker and retrieve the peer list. Tracker determines the layer to assign the peer and returns the list containing other peers at that layer. We first introduce two peer assignment methods.

1) *Allocation method with fixed amplifier layer size*. For most channels with moderate size, keeping track of the small amplifier layer would not pose much overhead on the tracker. One approach is to fix the size and the fractions of different types of peers at the amplifier layer based on the calculation in the previous section.

Algorithm 2: Peer Assignment with Fixed Amplifier Layer Size

```
input : New peer $n$
output: Supplier Set $\mathcal{S}_{sup}$
1  $i \leftarrow$ GetPeerType($n$)
2  $N_{max} = N_a p_i^a$
3  if GetPeerNumofType($i$) $< N_{max}$ then
4      AssignToAPLayer($n$)
5      mark $\leftarrow$ true
6  else
7      AssignToBaseLayer($n$)
8      mark $\leftarrow$ false
9  end
10 if $i$ equals $k$ and mark is true then
11     $\mathcal{S}_{sup} = \mathcal{S}_{sup} \cup \{n\}$
12 end
```

Algorithm 2 presents the corresponding peer assignment algorithm for the tracker. Tracker first sets the size of amplifier layer N_a and the fractions of different types of peers p_i^a. Then it determines the target number of each type of peers at the amplifier layer, $N_a p_i^a$ for peers with type i. When a new peer joins, tracker checks whether there is enough number of peers with the same type at the amplifier layer. if the number does no reach the target, the new peer will be assigned to the amplifier layer. If the new peer assigned to the amplifier layer is of the supplier type, the tracker puts it into the supplier set for later peer request from the base layer.

2) *Allocation method with adaptive amplifier layer size*. When the channel population changes dramatically, one should adaptively change the size of the amplifier layer. The fractions of each type peers at the two layers can be maintained close to target ratios derived in the previous analysis. For peers of type i, we can determine the percentage assigned to two layers repectively. The percentage of the ith type peers assigned to the amplifier layer should be

$$q_i^a = \frac{N_a p_i^a}{(N_a + N_b)p_i} = \frac{p_i^a/p_i}{1+\beta}. \tag{8}$$

Correspondingly the percentage of the ith type of peers assigned to the base layer is $1 - q_i^a$. Upon a new arrival of a ith type peer, the tracker assigns it to the amplifier layer with probability q_i^a, and the base layer otherwise. Then the amplifier layer can adapt its size and source bandwidth amplified ratio automatically with the current channel population. Furthermore, the tracker does not need to keep track of the amplifier layer status.

In practice, these two methods can be combined for peer assignment. At the system initialization stage, or when the amplifier layer size is smaller than a certain threshold, the tracker employs the allocation method with fixed amplifier layer size to stabilize the bandwidth contribution from the amplifier layer. Otherwise the latter one could be applied with more flexibility and scalability.

A traditional tracker needs to record the information of all peers and respond to peer list request at any time. In addition to the tracker, peers can also retrieve other peer information from their neighbors [5]. A lightweight tracker only needs to determine the layer to be assigned and return initial peer list when a peer just joins the system. Therefore the source can also play the role of tracker. Meanwhile it can only record part of the base layer peer information to save memory cost.

4.4.2 *Adaptive Management against Peer Dynamic*

Peers come and go frequently in P2P streaming systems. P2P streaming systems have to be designed to be robust against peer churn. With the robustness of the mesh-based streaming, the LPS architecture can be maintained well in face of peer churn.

1) *Cross-Layer Connection Establishment.* Suppliers at the amplifier layer push video streaming to receivers at the base layer through cross-layer connections. A peer assigned to the amplifier layer maybe chosen as a supplier by the previous assignment algorithm. The base layer receivers can be determined according to certain policy or property, for example, peer bandwidth in this implementation. Placing large bandwidth peers near the root can increase the fan-out degree of the date delivery tree and shorten the chunk delays. If a peer assigned to the base layer meets the criteria, the tracker adds one supplier IP address to its peer list. Then the peer can establish a direct connection with the supplier at the upper layer to download a whole video stream. If a supplier has reached its maximum contribution level (λ copies), it then requests the tracker to remove it from the supplier set. If a supplier, that was fully loaded and removed from the supplier set before, has available bandwidth to contribute again due to the departure of its receiver at the base layer, it can report to the tracker and to be re-inserted into the supplier set.

2) *Node Promotion upon System Imbalance.* The suppliers at the amplifier layer would be generally fully-utilized, in that the size of base layer is far larger than that of the amplifier layer. With the continuous arrivals of new peer, one available supplier would be quickly allocated to one of the newly joined peers at the base layer. In addition, the vacancy of the amplifier layer caused by peer departure can be easily filled by the new peers assigned provided that peers continuously come and go. In the case of batch peer departures, a large number of peers leave the system almost at the same time, thereby possibly not enough number of peers remain in the amplifier layer and the system may lose resource balance. To handle such unusual cases, some *promotion* procedure can be conducted as follows.

The peers at the base layer are allowed to be promoted to the amplifier layer, hence the system resource can be redistributed. However, the playback delay of peers have been elaborately differentiated and peers at different layers are supposed to have different playback progresses. To resolve the conflict of synchronous download among peers with asynchronous playback progress, we resort to a VOD-like buffer implementation. *Active buffer window* denotes the portion of buffer in use for smoothing peer playback progress. Figure 4 presents a snapshot of peer buffers. Peers at the

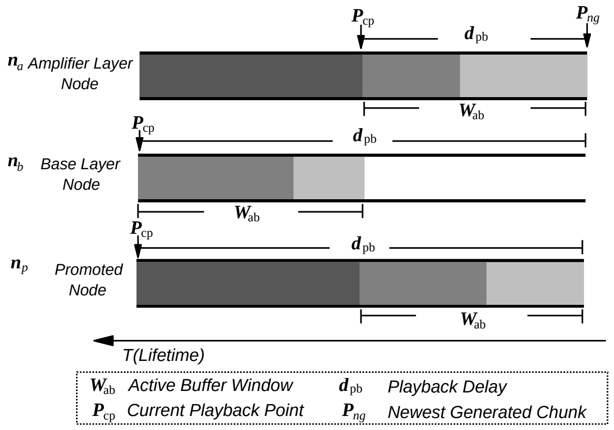

P_{cp}
d_{pb}
P_{ng}
n_a Amplifier Layer Node
W_{ab}
n_b Base Layer Node
n_p Promoted Node
T(Lifetime)
W_{ab} Active Buffer Window
d_{pb} Playback Delay
P_{cp} Current Playback Point
P_{ng} Newest Generated Chunk

Figure 4: Buffer Snapshot of Different Peers

amplifier layer n_a have smaller playback delay and download fresh chunks first. In this setting, the active buffer window $W_{ab} = W/2$, where W denotes the homogeneous buffer size of all peers. The base layer peers n_b would be only interested in chunks within their own active buffer window, leaving the latter half of buffer empty (In practical implementation, the buffer size of base layer peers can be set only to be $W/2$ to prevent memory waste). As for the amplifier layer peers, the content after playback would be kept available in their buffers for an additional $W - W_{ab}$ seconds instead of being removed immediately. Those chunks will be sent to the promoted nodes to speedup their buffering progress during the promotion process. After nodes n_p are promoted, they maintain their playback progress and delay unchanged, while the downloading would be synchronized with other peers at the amplifier layer gradually. And the active buffer windows of newly promoted peers would also move forward since other peers at the amplifier layer can serve as seeds for the content of previous active buffer window. In this way, the promoted peers from base layer and the original peers at amplifier layer can coexist with synchronized download progress while with asynchronous playback progress.

Once tracker detects such system imbalance, it can select residual suppliers randomly and issue promotion protocol messages to them. The selected suppliers ask the corresponding receivers to broadcast a promotion message through limit-hop flooding at the base layer. The base layer peers in the flooding range which meet the criteria, such as bandwidth, then request new neighbor list from tracker, tear down the old connections and join the amplifier layer as a new arrival peer. The tracker can control the number of the promotions by setting the frequency of promotion messages.

4.4.3 *Collaborative Push/Pull Data Delivery*

Within each layer, peers broadcast the data availability information via buffer-maps and pull data from each other. The source would push content proactively in the way introduced in Section 3. Suppliers also push content to their receivers at the base layer. Generally, we can set the playback delay gap between the two layers to be the active buffer window size of peers at the amplifier layer. Then a supplier can directly push the content just played to its receivers. As illustrated in Fig. 4, the supplier pushes playbacked content before they fall outside of the active buffer window region. The receivers at the base layer can receive the whole content if the corresponding supplier has no data loss. Therefore suppliers act as seeds for the base layer. The receivers at the base layer push content directly to their neighbors to accelerate the distribution process. They can also complementarily pull from their neighbors in case certain chunks are not received from their suppliers.

4.4.4 Practical Implementation Considerations

In practical implementation, the information of peer type can be piggybacked in the initial register message when the peer join the system. Generally the access type of peer implies the upload capability the peer has. In light of that peer available bandwidth may be dynamic sharing with other network applications, tracker can monitor the playback performance and adaptively increase the resource distribution of the amplifier layer accordingly. The basic LPS architecture has two layers. It is adequate enough to handle a reasonably large-scale video streaming. To further increase the scale, two-layer LPS architecture can be extended to multiple layers. The base layer can easily allocate certain number of peers to drive another layer with even larger playback delay. Powered by the same mechanism, multiple layers can be linked together. Streaming at each layer is driven by multiple proxies from the layer immediately above to achieve premium streaming performance.

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed strategies using extensive simulations and experiments. With packet-level P2P streaming simulations driven by traces from real systems and experiments on the PlanetLab , we demonstrate that the proposed strategies can greatly enhance the capability of bandwidth constrained source in driving large-scale streaming.

5.1 Simulation Setting

We developed a packet-level event-driven simulator in C++ to examine the performance. Our simulator adopts the architecture of the simulator engine of [12], which simulates the end-to-end latency between peers using real-world latency measurement results. We employed two 4-CPU servers to accelerate the simulations.

We follow the common consumption that peer download bandwidth is large enough and the bottlenecks happen only at uplinks in the edge access networks. There are three DSL types of peers with bandwidth 1Mbps, 384kbps and 128kbps. And the percentages of the three types of peers are 0.28, 0.40 and 0.32 respectively. The video streaming rate is 400kbps and the size of each chunk is $1,250$ Bytes. The source bandwidth is set to be 420kbps as default, slightly higher than the streaming rate in order to handle the possible signaling tasks. The resource index of the whole system is $\rho = 1.2$. By default peers try to maintain the neighbor size to be 15. In LPS, the size scale ratio of the amplifier and base layer is set to be $\beta = 19$. We pick the peers with 1Mbps bandwidth to be the supplier at the amplifier layer, which contributes one copy of the streaming rate to the base layer, i.e., $\lambda = 1$. Except for the trace-driven simulation, the simulation duration is set to be 300 seconds and data from the first 60 seconds, when the system might be unstable, are not included in the statistics.

5.2 Performance Metrics

The following performance metrics are investigated to evaluate the system in the simulations.

1. *Delivery Ratio*: In live streaming, chunks arriving after their playback deadlines are useless and dropped. The *delivery ratio* is defined as the number of chunks that arrive in time over the total number of chunks that the peer should receive. With homogeneous chunk size, this directly reflects the user playback quality, also the efficiency of the system distribution process.
2. *Bandwidth Utilization*: The *bandwidth utilization* shows how efficiently the system scheduling utilizes the peer resource. Higher bandwidth utilization means more resource would be invested to speed up the distribution process. It is calculated as the ratio between peers' upload rate and peers' upload capacity.
3. *Chunk Delay*: As discussed in Section 4.1, the *chunk delays* measure how fast the distribution process delivers chunks. It is computed as the difference between the chunk receive time and the time when it is firstly delivered to the network.
4. *Chunk Propagation Hop-count*: This measures the path length chunks traverse in the network to reach peers. A peer increases the hop count field of a chunk before it forwards the chunk to its neighbors.

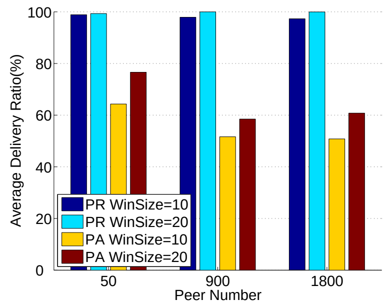
To facilitate the comparison, the chunk delay and hop count for the base layer in LPS would be calculated based on what chunks experience only within the base layer.

5.3 Numerical Results

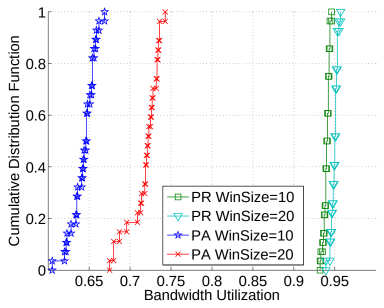
5.3.1 Proactive vs. Passive Source

First, we compare the system performance of the proactive source (PR) and the passive source (PA) schemes under various peer buffer window sizes and overlay network scales. We increase the network scale from 50 to 900 and 1800, and change the peer buffer window from 10 seconds to 20 seconds. Fig. 5(a) presents the delivery ratios of the two schemes under different settings. We can observe that the delivery ratio drops as the network scale increases. In the passive mode, the source is overwhelmed by peer requests and has difficulty in delivering content in time. With small source bandwidth 420kbps, only slightly higher than the streaming rate 400kbps, the average delivery ratio of all peers is always kept below 70% with 10 second buffer window. Increasing the buffer window to 20 seconds improves the delivery ratio only slightly. On the contrary, a PR source pushes fresh content without delay. This guaranteed in-time broadcast of fresh content increases the chunk diversity which enhances the distribution efficiency in return. When the peer buffer window is 20 second, all chunks can be received before their playback deadline. Even with only 10 second buffer window, the average delivery ratio is also above 97%. Fig. 5(b) shows the bandwidth utilization of various peer types when the network scale is 1800. PR achieves high bandwidth utilization of more than 90% at both buffer window settings. In contrast, the peer bandwidth utilization with passive source is low and longer buffering time helps improve the utilization.

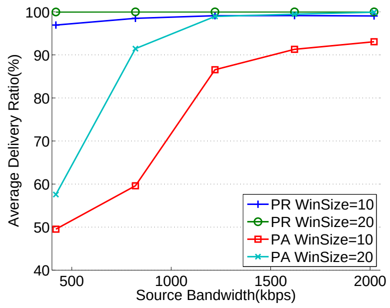
To examine how much source bandwidth is adequate to achieve universal streaming, we study the system performance with various source bandwidth settings. The total number of peer is $2,000$, a moderate network size, in the simulations. Fig. 5(c) shows the average delivery ratio is improved as the source bandwidth increases. For the PA scheme, with longer chunk download deadline allowed by large buffer window size (20s), the average delivery ratio can approach 100% when the source bandwidth is larger than 1.2Mbps. However, the inefficient scheduling of PA still confines the enhancements to some extent. With 10 second buffer window, the average delivery ratio can only reach 93% even when the source bandwidth is increased to 2Mbps. We can observe the PR strategy can greatly enhance the performance with very limited bandwidth. The average delivery ratio has already reached 97% when the source bandwidth is 420kbps and the buffer window size is 10 second. All peers can approach universal streaming under other settings.



9(a) ??? Impact of Network Size



9(b) Bandwidth Utilization



9(c) Impact of Source Bandwidth

Figure 5: Performance Comparison between Proactive Source and Passive Source

5.3.2 *Performance of the LPS Architecture*

To understand the effectiveness of LPS, we vary the source bandwidth amplified ratio and observe the corresponding performance. In this set of simulations, there are $2,000$ peers in total. With $\beta = 19$, there are 100 peers at amplifier layer and $1,900$ peers at base layer. We set the peers with 1Mbps bandwidth to be the suppliers and the corresponding distribution at the amplifier layer is $p_k^a = 0.8$. In terms of the distribution, the resource index of the amplifier layer $\rho_a \approx 1.3$ when all the suppliers contribute one copy of the streaming to the base layer. Here we define the control parameter, *cross layer factor* ϕ. We will add $1/\phi$ potential suppliers to the supplier set at amplifier layer. $1/\phi$ percentage of the suppliers at amplifier layer would contribute one copy of streaming to the base layer. Given $\phi = 1$, all suppliers contribute, while half of suppliers would contribute with $\phi = 2$, so one and so forth.

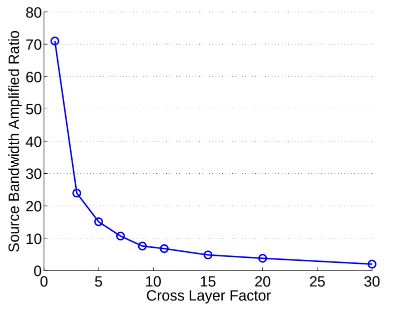
Fig. 6(a) shows the actual average number of streaming copies the amplifier layer contributes to the base layer with different ϕ. Naturally it decreases as ϕ increases. Compared with the theoretical calculation $\alpha_{amp} = N_a p_k^a/\phi$, the actual amplified ratio is slightly less, due to the supplier bandwidth expenditure competition among the inner-layer data pull request and inter-layer data push. Nevertheless the amplified ratio reaches 70 when all suppliers contribute to the base layer ($\phi = 1$), i.e., the source bandwidth has been amplified 70 times. An aggregate proxy source bandwidth of 28Mbps will be used to the base layer with 95% of peers.

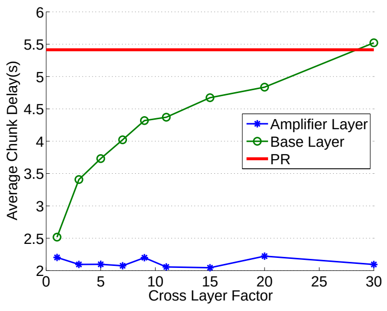
The multiple suppliers at the amplifier layer can greatly shorten the delivery path length and decrease the chunk delays for peers at the base layer. Fig. 6(b) and 6(c) show the average chunk delay and hop count with different ϕ setting. At the amplifier layer, the chunk delay and hop-count are kept small, benefiting from the small network scale even though the source bandwidth is limited. The bandwidth amplified ratio α_{amp} decreases as ϕ increases. For the base layer, the chunk hop count increases from around 2.7 to 5.2 and the delay increases from 2.5 to 5.5 seconds correspondingly. For the convenience of comparison, we also plot the chunk delay and hop count of the PR scheme with the same network scale and source setting. It is only comparable at $\phi = 30$ when the aggregate supplier bandwidth contributed to the base layer is the least, and almost equals to the original source bandwidth.

Next we investigate the capability of LPS to drive large-scale streaming. We run simulations for LPS with different network sizes. In all simulations, we use the same setting of the LPS scheme and $\beta = 19$. The amplifier layer size adapts correspondingly and the peers at the amplifier layer can always achieve nearly universal streaming. Fig. 7(a) shows the average delivery ratio of the different strategies with different buffer window sizes. With 10 second buffer window, the LPS can promisingly achieve above 99% delivery ratio even when the total peer number reaches $12,000$. Furthermore, the LPS is able to achieve satisfactory performance with smaller buffer window size of 6 seconds. The average delivery ratio always remains above 96% under the setting. However, as the network scale increases, the delivery ratio with PR scheme drops inevitably below 93% with 10 second buffer window. The performance deteriorates with smaller buffer window size (8s). The delivery ratio drops even below 84% eventually. It turns out that the PR scheme cannot sustain the system when the system scale is large, although it outperforms the PA scheme greatly. Fig. 7(b) illustrates the evolution of chunk propagation hop count as the network scale increases. The size of amplifier layer increases as the total number of peers since the ratio β is fixed. Therefore the chunk propagation hop counts of the LPS layers and PR network both increase. On the other hand, the source bandwidth amplify ratio is also increased. The propagation hop count of the base layer increases slightly, always below 3. In other words, chunks commonly need fewer than three hops to reach every peer from the suppliers at base layer. This shows LPS not only is able to support large-scale network but also adapts well to the network scale. It is noticeable that in this set of simulations, the buffer window size of amplifier layer peers is set to be 6 seconds only. Thus the playback delay of base layer is only 12 seconds as the buffer window size of base layer peers is 6 seconds. It is the enhanced distribution efficiency that enables users to still get satisfactory playback delay in LPS.

5.3.3 *Comparison of Start-up Delays*

As discussed in [22], different implementations lead to different start-up delays. Generally, efficient distribution process enables the peers grab chunks quickly when they first join the system. Here we compare the peer buffer status with different strategies during the start-up phase. Peer buffer window size is set to be 10 seconds. When new peers join the system, they are allowed to download from the middle of other peers' buffer window, i.e., the last 5 seconds content is downloadable. Since the download point has been synchronized with the source, the download window increases as time passes. **the following sentences are not very clear to me** We will record the percentage of the buffer window that has been filled after 5 seconds, at that time the buffer window has been increased to 10 seconds, same as other peers. $2,000$ peers will join the system first at the beginning, then another $2,000$ peers will gradually join from time 100 second to 200 second. Fig. 7(c) shows the CDF of the percentage of filled buffer window after the first 5 seconds for those $2,000$ peers who randomly join the system from time 100 to 200 second. We can observe that during the start-up phase, the peers can fill up, in average, 76% of the buffer window in LPS,


9(a) Source Bandwidth Amplified Ratio


9(b) Average Chunk Delay

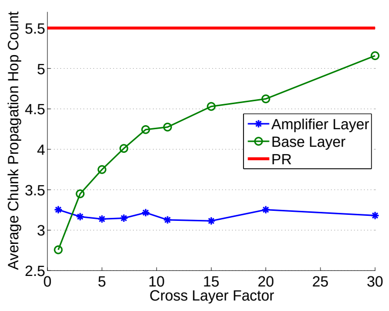
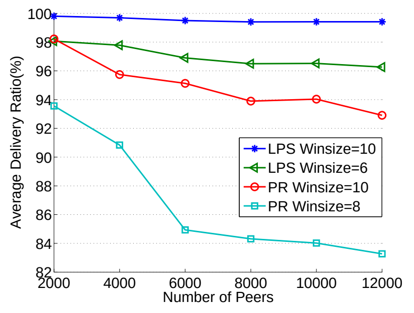
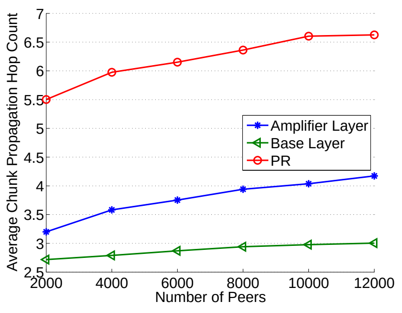

9(c) Average Chunk Propagation Hop Count

Figure 6: Understand the LPS architecture


9(a) Average Delivery Ratio


9(b) Average Chunk Propagation Hop Count

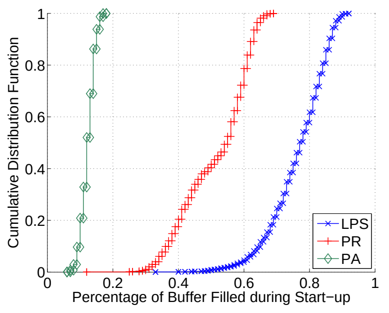

9(c) Start-up Delay

Figure 7: Comparison between the LPS and PR schemes

while 51% and 12% for PR and PA respectively. This promising result also demonstrates that LPS can improve the start-up performance, and the start-up delay can be shorten to 5 seconds for all peers if they are configured to start the playback after buffering 5 seconds worth of content.

5.3.4 *Trace-driven Evaluation*

To examine the system performance under peer churn, we evaluate the strategies driven by peer session traces taken from a measurement study of a real P2P streaming system [5]. The trace records the peer arrival and departure information of one popular channel on Oct.16, 2006. From this one-day trace, we pick the data from $9:00$AM to $9:30$AM period, the corresponding peak time of that channel. In every minute of this half hour trace, there are in average around 148 new peer joins and 135 peer departures. The minimum and maximum aggregate number of online peers are around $3,200$ and $4,000$ respectively. In this trace-driven simulation, the source bandwidth is 450kbps, still slightly higher than the streaming rate of 400kbps. And we fix the amplifier layer size to be 200 in LPS. The peer buffer window sizes of the schemes are all set to be 10 seconds. Fig. 8 shows the evolution of the delivery ratio of schemes with peer dynamics. The number of peers changes dramatically, implying intense peer churn throughout the simulation. We can observe the performance of LPS is very steady and the chunk delivery ratio remains above 98%. On the contrary, the delivery ratio of the PR scheme oscillates with large amplitude and finally the delivery ratio drops around 80%. The above results show the LPS scheme is very robust against peer churn. With the peer assignment algorithm in Section 4.4.1, the amplifier layer can be well maintained in face of frequent peer arrivals and departures.

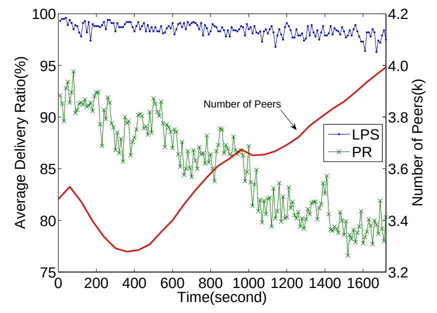


Figure 8: Trace-driven Simulation

To further examine the system performance in real Internet environments, we developed a prototype and conducted experiments on the PlanetLab. In the small scale experiment with around 300 nodes (constrained by the simultaneously available online nodes of PlanetLab), we still drive the system with the peer arrivals and departures data of the above trace while scaling down accordingly. At the beginning, around 180 peers join the system simultaneously. After 200 seconds, peers arrive and leave the system according to the scaled trace records. Every 30 seconds, we either randomly select a certain number of idle nodes to let them join the system, or randomly select a certain number of online peers to let them leave the system. Hence the number of online peers jumps up and down suddenly every half minute. The minimum and maximum online peers are around 140 and 240 respectively. The source pushes only one copy to the amplifier layer with 360kbps bandwidth, slightly higher than the streaming rate 320kbps. The amplifier layer size

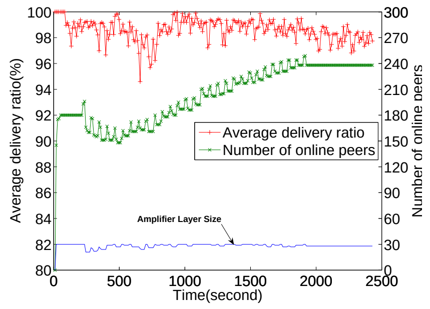


Figure 9: Experiments on PlanetLab

is fixed to be 30 and each of the upper suppliers only pushes one copy to the base layers. And the sizes of all peer buffer map windows are set to be only 8 seconds. Fig. 9 shows the evolution of the average delivery ratios of all peers as new peers join the system driven by the trace. The system can maintain high delivery ratio (always around 98%) with bandwidth constrained source. Besides, the amplifier size maintains stable with peer churn.

6. CONCLUSION

In this paper, we studied the strategies for users to directly broadcast User-Generated live video to a large number of audiences without any server support. First we discussed the impact of source chunk scheduling on the system performance. We proposed a proactive source scheme to push rarest-first contents. Then we proposed a novel layered P2P streaming architecture (LPS) to amplify the source bandwidth by introducing peer playback delay differentiations deliberately. We developed a detailed mesh-pull based LPS implementation, which consists of tracker-based peer assignment, dynamic peer management and cross-layer push-pull data scheduling. Through extensive trace-driven simulations and experiments, we demonstrated that the proposed strategies enable a “weak" source with upload bandwidth slightly higher than the encoded video rate to drive a P2P streaming session with tens of thousands of peers. By sacrificing a little bit the playback delays for peers at the base layer, the proposed LPS architecture can achieve short start-up delays and low video chunk loss ratios for all peers at both the amplifier layer and the base layer.

7. REFERENCES

[1] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, “I Tube, You Tube, Everybody Tubes: Analyzing the World’s Largest User Generated Content Video System,” in *Proceedings of Internet Measurement Conference*, 2007.

[2] C. Huang, J. Li, and K. W. Ross, “Can Internet Video-on-Demand be ProÞtable?” in *Proceedings of ACM SIGCOMM*, 2007.

[3] Unkown, “PPLive Homepage,” http://www.pplive.com.

[4] PPStream, “PPStream Homepage,” http://www.ppstream.com.

[5] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, “A measurement study of a large-scale P2P IPTV system,” *IEEE Transactions on Multimedia*, 2007.

[6] Y.-H. Chu, S. G.Rao, and H. Zhang, “A case for end system multicast,” in *Proceedings of ACM SIGMETRICS*, 2000.

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

[8] V. Venkataraman, K. Yoshida, and P. Francis, “Chunkyspread: Heterogeneous unstructured end system multicast,” in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2006.

[9] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, “Chainsaw: Eliminating trees from overlay multicast,” in *IPTPS*, 2005.

[10] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, “DONet/CoolStreaming: A data-driven overlay network for live media streaming,” in *Proceedings of IEEE INFOCOM*, 2005.

[11] N. Magharei and R. Rejaie, “PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming,” in *Proceedings of IEEE INFOCOM*, 2007.

[12] M. Zhang, Q. Zhang, L. Sun, and S. Yang, “Understanding the power of pull-based streaming protocol: can we do better?” *IEEE Journal on Selected Areas in Communications*, 2007.

[13] F. Wang, Y. Xiong, and J.Liu, “mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast,” in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2007.

[14] M. Wang and B. Li, “Lava: A Reality Check of Network Coding in Peer-to-Peer Live Streaming,” in *Proceedings of IEEE INFOCOM*, 2007.

[15] Y. Liu, “On the minimum delay Peer-to-Peer video streaming: how realtime can it be?” in *Proceedings of ACM MULTIMEDIA*, 2007.

[16] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, “P4P: Provider Portal for Applications,” in *Proceedings of ACM SIGCOMM*, 2008.

[17] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, “Utility Maximization in Peer-to-Peer Systems,” in *Proceedings of ACM SIGMETRICS*, 2008.

[18] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, “Substream Trading: Towards an Open P2P Live Streaming System,” in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2008.

[19] R. Kumar, Y. Liu, and K. Ross, “Stochastic fluid theory for P2P streaming systems,” in *Proceedings of IEEE INFOCOM*, 2007.

[20] C. Wu, B. Li, and S. Zhao, “Multi-channel live p2p streaming: Refocusing on servers,” in *Proceedings of IEEE INFOCOM*, 2008.

[21] N. Magharei, R. Rejaie, and Y. Guo, “Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches,” in *Proceedings of IEEE INFOCOM*, 2007.

[22] B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu, and X. Zhang, “Inside the new coolstreaming: Principles, measurements and performance implications,” in *Proceedings of IEEE INFOCOM*, 2008.