LayerP2P: Using Layered Video Chunks in P2P Live Streaming

Zhengye Liu, Yanming Shen, Keith W. Ross, *Fellow, IEEE*,
Shivendra S. Panwar, *Senior Member, IEEE*, Yao Wang, *Fellow, IEEE*,

***Abstract*—Although there are several successful commercial deployments of live P2P streaming systems, the current designs (i) lack incentives for users to contribute bandwidth resources, (ii) lack adaptation to aggregate bandwidth availability, and (iii) exhibit poor video quality when bandwidth availability falls below bandwidth supply. In this paper, we propose, prototype, deploy and validate *LayerP2P*, a P2P live streaming system that addresses all three of these problems. LayerP2P combines layered video, mesh P2P distribution, and a tit-for-tat-like algorithm, in a manner such that a peer contributing more upload bandwidth receives more layers and consequently better video quality. We implement LayerP2P (including seeds, clients, trackers, and layered codecs), deploy the prototype in PlanetLab, and perform extensive experiments. We also examine a wide range of scenarios using trace-driven simulations. The results show that LayerP2P has high efficiency, provides differentiated service, adapts to bandwidth deficient scenarios, and provides protection against free-riders.**

***Index Terms*—Peer-to-peer streaming, layered video.**

I. Introduction

With the widespread adoption of broadband residential access, P2P live video streaming has become a popular service in the Internet. Recently, several P2P live video systems have been successfully deployed, supporting tens of thousands of simultaneous users in a single channel, with stream rates between 300 kbps to 1 Mbps. These systems include CoolStreaming [1], PPLive [2], [3], PPStream [4], UUSee [5] and many more. All of these systems use a chunk-based mesh-pull design with single-layer video.

Although the success of these P2P live video systems shows the potential of efficiently broadcasting live video content to a large number of users in the Internet, P2P live video streaming is still in its early stages. These systems have several critical design problems, limiting their robustness, scalability, and performance:

- **Lack of incentives:** Peers have heterogeneous upload bandwidths and exhibit a wide range of altruistic behavior for contributing upload bandwidth. In existing P2P live video systems, peers all receive the same video quality no matter how much upload bandwidth they contribute to the system. To build a robust and scalable P2P system, it is critical to provide incentives to reward peers that contribute more.
- **Lack of adaptation to bandwidth availability:** In P2P live video systems, both the system supply (the aggregate upload bandwidth of the peers) and the system demand (the aggregate download rate of the peers) fluctuate. This long-term bandwidth fluctuation is mainly due to peer churn. When the system operates in a bandwidth-deficient regime, where the system supply is less than the system demand, the user experience will be significantly degraded.
- **Severe degradation of video quality:** Packet loss in P2P streaming is not only caused by transmission loss (e.g., due to network congestion), but also by the lack of bandwidth and/or content at the supplying peers. With single layer video, any lost packet can lead to significant degradation in decoded video quality, due to spatial-temporal prediction and entropy coding in the encoded video. Typically a video is coded into groups of pictures (GOP) with the first frame of each GOP coded without referencing previous frames (known as the I-frame) and all following frames coded with references to previous frames. A lost packet not only affects the video frame it belongs to, but also affects the successive frames due to error propagation. This induces severe video quality degradation.

In this paper, we propose, prototype, deploy and validate **LayerP2P**, a P2P live streaming system that simultaneously addresses all of the above three problems. LayerP2P applies layered video on mesh-pull P2P live streaming systems. With layered coding, a video is coded into layers with nested dependency: a higher layer refines the video generated by lower layers. More received layers provide better video quality.

In the past, the coding efficiency of layered coding was significantly lower than that of single layer coding, which has hindered its deployment in practical systems. However, in recent years the coding efficiency of layered video has been significantly improved. For example, the newly established

This work was supported by the National Science Foundation under Grant No. CNS-0435228 and the New York State Center for Advanced Technology in Telecommunications (CATT) at Polytechnic Institute of New York University, Brooklyn, NY, USA.

Z. Liu, S. S. Panwar, and Y. Wang are with the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY 11201 USA (e-mail: zhengye@vision.poly.edu; yao@poly.edu; panwar@poly.edu).

Y. Shen was with the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY 11201 USA. He is now with Computer Science and Engineering Department at Dalian University of Technology, China (e-mail: shen@dlut.edu.cn).

K. W. Ross is with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, NY 11201 USA (e-mail: ross@poly.edu).

H.264/SVC (layered coding) achieves a rate-distortion performance comparable with H.264/AVC (single-layer coding), with the same visual reproduction quality typically achieved with at most 10% higher bit rate [6]. Real-time systems with H.264/SVC encoder and decoder have been successfully implemented [7]. Real-time decoders [8] that support H.264 temporal scalable coding have been made available in the public domain. With these advances, layered video is ready for deployment in practical video applications. We will show, in particular, that it can significantly enhance P2P live video systems.

LayerP2P has a different design philosophy compared with the existing P2P live streaming deployments. In LayerP2P, when the system has abundant bandwidth, where the average upload bandwidth supply is higher than the full video rate, every peer can receive all layers of the video and enjoy excellent quality. However, when the system is in a bandwidth-deficient state (e.g., due to too many peers with low upload bandwidth or too many free-riders), so that not all peers can receive the full video rate, a peer's received video quality is commensurate with its upload contribution to the system. The peers providing high upload contribution receive high video quality; the peers providing moderate upload contribution receive lower but still acceptable video quality; while the free-riders receive at most poor video quality. LayerP2P has the following key characteristics:

- **Built-in incentives:** With layered video, more received video chunks in the order of their importance lead to higher video quality. LayerP2P exploits this property, together with a tit-for-tat-like strategy, to provide incentives for uploading. Specifically, each peer measures its download rates from its neighbors, and reciprocates by providing a larger fraction of its upload rate to the neighbors from which it is downloading at higher rates. Using this mechanism, when the system is overloaded and cannot support all peers with the full video rate, the video quality a peer receives is commensurate with its upload contribution. LayerP2P therefore uses the following incentive principle: the more a peer contributes, the better its received video quality.
- **Adaptation to available upload bandwidth:** LayerP2P dynamically adapts the system demand to the system supply. As the system aggregate bandwidth supply evolves due to peer churn, LayerP2P automatically adjusts video quality for the individual peers. From the perspective of the overall system, aggregate bandwidth deficiency, a nemesis for single-layer designs, is thus largely avoided.
- **Graceful video quality degradation:** With LayerP2P, lost packets in an enhancement layer do not affect the decoding of lower layers. Our proposed chunk requesting and scheduling schemes give higher priority to more important layers.

We develop a prototype for LayerP2P, including seed implementation, client implementation, and tracker. Using actual layered encoded video, we deploy the prototype in PlanetLab and conduct extensive experiments. To further understand the system behavior with realistic peer dynamics, we also conduct trace-driven simulations by using the traces for peer dynamics from a real-world P2P live streaming system. We compare LayerP2P with two single-layer P2P live video streaming systems. Both the experiments with the prototype in PlanetLab and the trace-driven simulations show that: (i) when the average system upload bandwidth is higher than the full video rate, LayerP2P can efficiently use the system resource to provide good video quality to all peers; (ii) when the average system upload bandwidth is lower than the full video rate, LayerP2P provides differentiated services for different peers and also prevents free-riding. Compared with the corresponding single-layer video systems, for both scenarios, LayerP2P provides an improved video quality for cooperative peers.

The remainder of this paper is structured as follows. Section II sets the stage for LayerP2P by providing a brief overview of mesh P2P systems and modern layered video. Section III presents the design of LayerP2P. In Section IV and V, we describe our implementation and evaluate its performance in a PlanetLab deployment. In Section VI, we evaluate the performance of LayerP2P in a wider range of scenarios using trace-driven simulations. Section VII describes related work. We conclude in Section VIII.

II. Overview of P2P Streaming and Video Coding

Before describing LayerP2P in details, it is beneficial to first review P2P streaming and modern video encoding.

A. P2P Live Streaming Systems

The chunk-based mesh design is the most popular and successful design in P2P live streaming today, due to its robustness to peer dynamics, high network scalability, and simplicity. Most of the existing P2P live streaming systems adopt this design.

In a chunk-based mesh-pull delivery architecture for live video streaming, as shown in Figure 1(a), the source divides the encoded bit stream into video chunks and then disseminates the video chunks to a set of randomly selected peers. When a peer wants to view the video, it obtains a list of peers currently watching the video. This procedure can be implemented by using a tracker that maintains all peer information, or with a DHT or gossiping. After obtaining the peer list, the peer selects several peers as its neighbors and establishes neighbor relationships with them. A neighbor relationship can be a real TCP connection between two peers, or simply a conceptual relationship without a real TCP connection. The peers in the system are self-organized into a mesh overlay, as illustrated in Figure 1(b). A peer acts as a supplier when it sends video chunks to its neighbors, while it acts as a receiver when it requests video chunks from its neighbors. As shown in Figure 1(c), each peer caches video chunks and maintains an exchange window. Typically, the exchange window includes all chunks between the playback time of a particular peer t_p and the video encoding time t_e at the video source. Periodically, neighbors exchange buffer maps with each other, explicitly informing their available chunks in the exchange windows. A peer carefully schedules its needed chunks based on the buffer maps and requests them from its

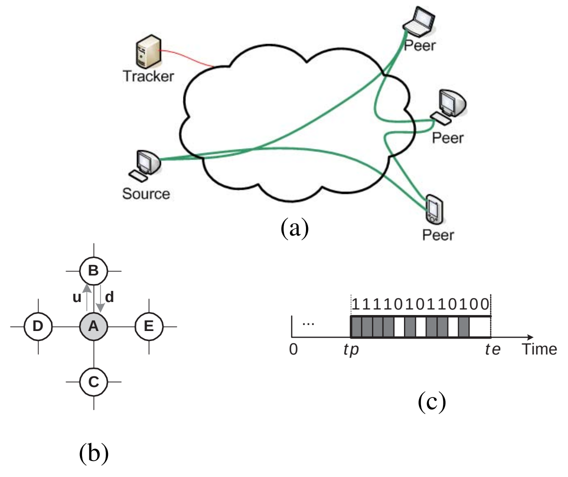

Fig. 1. Architecture of chunk-based mesh-pull systems. (a)Overall architecture; (b)Mesh overlay; (c)Buffer structure and buffer map.

neighbors. After receiving the requests, the neighbors serve the scheduled chunks to this peer based on some policy (with the most common practice being first come first serve for single layer video). In such a chunk-based design, each media chunk will be explicitly identified, requested, and scheduled.

B. Layered Video

Layered coding encodes a video into multiple layers with nested dependency: the base layer alone typically provides an acceptable basic quality, while a higher layer refines the video generated by lower layers. Layered coding is typically accomplished by providing multiple versions of a video either in terms of amplitude resolutions (called quality scalability or SNR scalability), spatial resolutions (spatial scalability), temporal resolutions (temporal scalability), frequency resolutions (frequency scalability or data partition), or combinations of these options.

In recent years, significant advances have been made in layered coding. H.264/SVC [9], the most recent scalable video coding standard, supports SNR scalability (coarse granularity scalability (CGS) and medium granularity scalability (MGS)), spatial scalability and temporal scalability. It provides the flexibility to encode a video into a large number (more than four) of layers. More importantly, coding efficiency of layered video has been significantly improved in H.264/SVC. Now H.264/SVC (layered coding) achieves a rate-distortion performance comparable with H.264/AVC (single-layer coding), with the same visual reproduction quality typically achieved with at most 10% higher bit rate [6]. The temporal scalable video coding can achieve an even higher coding efficiency than single-layer coding. Additionally, the video coding complexity, especially the decoding complexity, is well addressed in H.264/SVC. In [7], it is reported that a real-time H.264/SVC encoder and decoder have been successfully implemented, for different types of scalability. In particular, temporal scalable video is supported by both H.264/SVC and H.264/AVC due to its high video coding efficiency. FFmpeg[8] is an open source codec that can decode H.264 temporal scalable video in real-time.

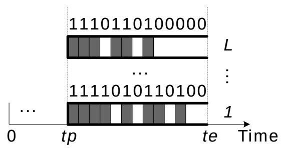

Fig. 2. Buffer structure of the layered system.

In LayerP2P, we do not make any assumption about the format of layered video. Any scalable or layered video coding scheme (temporal scalability, spatial scalability, SNR scalability, or any combination of them) can be incorporated into the LayerP2P framework. LayerP2P can work with either a few or many layers. Importantly, we do not require advanced rate control and rate partition mechanisms for creating layers, i.e., each layer can have an arbitrary bit rate. Furthermore, we do not require a specific GOP structure, video packetization, error resilience/concealment scheme, etc. As we will see in Section V, we use FFmpeg as the video decoder with its default configurations in our implementation, without any modification. We note that this loose requirement on the video coder is enabled by the chunk-based requesting and delivery architecture adopted by LayerP2P.

III. Design Philosophy

In this section we describe the design of LayerP2P, including neighbor establishment, incentive strategy, and video chunk scheduling algorithms. We highlight how the features of LayerP2P prevent free-riders.

A. Overview of LayerP2P

LayerP2P uses layered video (instead of single-layer video) on chunk-based mesh P2P live streaming systems. The video source encodes a video into L layers and each layer is sliced into packets, called *layer chunks (LCs)*. These LCs are distributed over a self-organized mesh overlay, as we describe subsequently. Similar to the existing single-layer systems (e.g. PPLive), when a peer joins in the system, it obtains a peer list from a tracker. The peer then selects a subset of peers from the list and forms neighbor relationships with them. In our design, peers categorize their neighbors into different types and treat them differently.

Unlike the single-layer video systems, in LayerP2P each peer maintains L buffers, one for each layer, with each buffer caching the LCs for its layer (see Figure 2). A peer's buffer map is a data structure that indicates which LCs it currently has. Each peer periodically exchanges its buffer map with its neighbors. Upon learning what LCs its neighbors have, a peer sends requests for its missing LCs. As discussed below, the peer carefully prioritizes the requests to maximize its received video quality. As a supplier, a peer may receive multiple requests from multiple neighbors. Based on a tit-for-tat like strategy, the supplier allocates larger fractions of its upload bandwidth to the neighbors who have higher upload contributions to the supplier. A peer may have neighbors who serve it with a low rate because of a lack of bandwidth, a lack of content, or an unwillingness to contribute. To obtain better neighbors, each peer modifies its neighbors periodically.

B. Neighbor Management

Neighbors are two peers that connect with each other to exchange data chunks. In our design, each peer classifies its neighbors into *initiators* and *receptors* and treats them differently. If peer A initiates the neighbor relationship by actively sending peer B a neighbor establishment message, then peer A is an initiator for peer B, while peer B is a receptor for peer A.

In P2P live streaming, each peer should have a sufficient number of neighbors to maintain the connectivity of the overlay. But, to limit the overhead, a peer should not have too many neighbors. Additionally, the system should avoid completely filling its connection slots, so that newcomers can get in. In our design, if a peer has less than $N_{\min}$ neighbors, it will actively seek new neighbors; if the peer has greater than $N_{\min}$ and less than $N_{\max}$ ($N_{\max} > N_{\min}$) neighbors, it passively accepts new neighbors, but will not actively seek new neighbors; if the peer has $N_{\max}$ neighbors, it will decline all neighbor establishment requests. In this manner, typically a peer has less than $N_{\max}$ but more than $N_{\min}$ neighbors, which maintains the connectivity of the overlay, while reserving room for newcomers. Like BitTorrent, in order to locate a better neighbor with higher uplink bandwidth and more content, in our scheme a peer periodically replaces the neighbor with the least contribution with a new peer.

C. Tit-for-Tat with Layered Video

BitTorrent is a remarkably popular file-distribution technology, with millions of users sharing content in hundreds of thousands of torrents on a daily basis. BitTorrent's incentive principle is as follows: a peer will get the file faster if it contributes more upload bandwidth to the torrent. This incentivizes users to upgrade their ISP access and/or increase the maximum upload rates (typically configurable) in their BitTorrent clients. BitTorrent provides this basic incentive using the celebrated tit-for-tat algorithm [10], in which peers trade blocks of content with each other. (Although several recent studies have shown that the tit-for-tat algorithm is not sufficient for preventing free-riders or fully incentivizing users [11], the algorithm has nevertheless been very successful in practice. If BitTorrent had been designed without a tit-for-tat algorithm, it almost surely would not have the success that it enjoys today.) Tit-for-tat effectively creates a differentiated service at the application layer, providing high-speed uploaders with short download times and low-speed uploaders with long download times.

Inspired by BitTorrent's incentive philosophy, we apply the tit-for-tat-like strategy on P2P live streaming systems. *We advocate a new incentive principle for live P2P streaming, namely, peers that upload more see higher quality video.* More specially, in our proposed system, each peer measures its download rates from its neighbors. A peer reciprocates to its neighbors by providing a larger fraction of its upload rate to the neighbors from which it is downloading at the higher rates. In this manner, a peer with higher upload contribution is likely to be rewarded with more LCs, and hence more layers and better quality. This strategy is implemented through the supplier side scheduler described below.

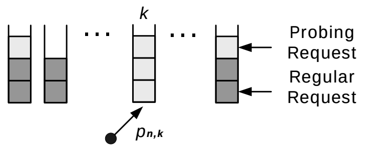

Fig. 3. Request queue structure at a supplier.

D. Supplier Side Scheduler

As a supplier, a peer receives LC requests from multiple neighbors. The peer must determine which requests should be served first and how to allocate its available uplink bandwidth to its neighbors. In our design, a peer will upload more to the neighbors from which it downloads more. To this end, a supplier maintains a different request queue for each receiver, as shown in Figure 3. For a particular receiver, the queue is first-in-first-out, where the supplier serves the requests in the order that the requests are received. (In Section III-E, we provide a refinement of the supplier side scheduler.) The supplier transmits one requested LC to one receiver at a time. Each request has a deadline to be served. If the supplier cannot serve a request before its deadline, it simply removes this request from the request queue and does not serve this request. More details about the request deadline will be described in Sec. III-F.

The supplier determines which receiver should be served depending on the receiver's contribution to the supplier. At any one time, the supplier randomly selects a receiver to serve. Let $p_{n,k}$ denote the probability that peer n selects the receiver k. $p_{n,k}$ is determined as follows:

$$p_{n,k} = \frac{I_{n,k}(d_{n,k} + \epsilon)}{\sum_{i \in \mathcal{K}_n} I_{n,i}(d_{n,i} + \epsilon)}, \qquad (1)$$

where $\mathcal{K}_n$ is peer n's set of neighbors, $d_{n,i}$ is the rate at which peer n is receiving from peer i, and ϵ is a small positive number. $I_{n,i}$ equals 0 if the request queue of receiver i is empty, and equals 1 otherwise. ϵ ensures that all of peer n's neighboring peers will be considered even if its $d_{n,k} = 0$. In summary, a receiver that uploads more to the supplier has a higher probability of being served, consuming a larger share of the supplier's uplink bandwidth.

Note that supplier always sends an LC as long as one of its queues is non-empty and if it has surplus upload bandwidth. In particular, a receiver with a low $d_{n,i}$ will be served when the other receivers do not have outstanding requests. Therefore, peers with a low upload bandwidth can receive a high video rate when the system is underloaded.

When two peers, say peer k and n, newly establish a neighbor relationship, they treat each other as if they have downloaded from each other in the recent past. But the two new neighbors treat each other unequally with different initial download rates. The initiator peer k treats its receptor peer n

as if n is providing a large download rate d_l, and therefore allocates a large share of upload bandwidth to n. On the other hand, the receptor peer n treats its initiator peer k as if it has a small download rate d_s, and therefore allocates a small share of upload bandwidth to it. This strategy is similar in spirit to the "optimistic unchoke" in BitTorrent. The neighbors, especially the initiators, are generous to each other. A small d_s prevents a free-rider from obtaining high download rates by actively adding a large number of neighbors. This will be described in more details in Sec. III-F.

E. Receiver Side Scheduler

As a receiver, a peer may have multiple missing LCs, and for a given LC there may exist multiple neighbors that have it. The peer needs to determine how to request these LCs to maximize its received video quality.

In single layer video systems, it has been shown that a simple random scheduling scheme is sufficient to achieve a high system throughput (i.e., high usage of available upload bandwidth of peers) [12], [13]. This scheme works as follows. Peers request their missing chunks in rounds. In each round of T seconds, the receiver requests all the chunks that it does not have but are available at its neighbors. The receiver requests these missing chunks in a random order without prioritizing them. For a particular chunk, the receiver randomly selects a supplier to serve this chunk if it is available from more than one supplier. The receiver groups all chunk requests scheduled to a particular supplier into one request message. In the next round, the receiver schedules the missing chunks which are newly available in its neighbors, together with the previously scheduled-but-not-served chunks. To avoid sending duplicated chunks, at the supplier side, each chunk request has a serving deadline. If the supplier cannot serve a chunk request within time $T - \tau$ (where τ represents the round-trip delay between the receiver and the supplier) from when it receives the chunk request, it simply removes this chunk request from its request queue.

Although the simple random scheduling scheme just described works well for single-layer video, it cannot be directly applied to layered video. This is because in layered video systems, a higher downloading rate does not necessarily translate to a better video quality. A receiver peer needs to prioritize the LC requests based on their importance for the reconstructed video. In general, if the available downloading rate of a receiver is less than the full video rate, it should only request the LCs from the lower layers such that the aggregate rate is below its downloading rate. A receiver faces a dilemma: on the one hand, if it requests too aggressively for higher layers, the LC requests for lower layers may not be able to be served before its playback deadline (which is different from the $T - \tau$ deadline in this round); on the other hand, if it requests too conservatively, it may not fully utilize the potential bandwidth of its supplier.

To solve this problem, LayerP2P uses a prioritized random scheduling algorithm tailored for layered video. With such an algorithm, the requests for different LCs are categorized into two types, namely, regular requests and probing requests. A peer expects regular requests to be served on time with high probability, and expects the probing requests to be served when the suppliers have surplus upload bandwidth allocated to this peer. For a particular peer n, the LC requests for the layers lower than or equal to a threshold l_n are designated to be regular requests, while the LC requests for higher layers are designated to be probing requests. l_n is determined by the expected available downloading rate of peer n. Let

$$u_{n,k} = \frac{d_{n,k} U_n}{\sum_{i \in \mathcal{K}_n} d_{n,i}}, \qquad (2)$$

where $u_{n,k}$ is the upload bandwidth allocated from peer n to peer k. It has been proven [14] that, if each peer performs the pair-wise proportional bandwidth allocation to its neighbors, then the available download rate of peer n (denoted as D_n) asymptotically equals its upload rate (denoted as U_n). It is not difficult to see that the supplier scheduling algorithm in Section III-D follows the pair-wise proportional bandwidth allocation, when the system is in the saturated regime, where each peer always has requests to serve. Thus, we set l_n to the largest layer index l for peer n, where $R(l) \leq U_n \leq R(l+1)$ and $R(l)$ denotes the video rate up to layer l.

The regular requests (for layers 1 to l_n) are assigned to different suppliers based on the previously described random scheduling algorithm for single layer video, without prioritization among different layers (from layer 1 to layer l_n). The probing requests (for layer l_{n+1} to L) are sent to the suppliers layer by layer, beginning with layer l_{n+1} and ending with layer L. For a particular layer, the probing requests are scheduled based on the above random scheduling algorithm. In summary, for each receiver, the regular requests have strictly higher priority over the probing requests, and the probing requests from lower layers have higher priority than those from higher layers. The reason that we do not prioritize the regular requests by layers is to preserve the randomness of the requests, thereby increasing the overall system throughput. The receiver scheduling algorithm is summarized in Algorithm 1.

F. Free-rider Prevention

A free-rider is a peer who intends to enjoy the system's services without contributing. As observed in P2P file downloading systems, free-riding is a prevalent problem which can significantly degrade the system performance. LayerP2P has several features that prevent free-riding. With the pair-wise proportional bandwidth allocation, a free-rider can only obtain a small share of upload bandwidth from its neighbors in an overloaded situation, no matter how many neighbors this peer has [14]. This limits its download rate and received video quality. With the periodic neighbor adaptation, a free-rider experiences unstable neighbor relationships which normally leads to unstable video quality. After a free-rider is dropped by its neighbors, it may switch to other new neighbors for free-riding. However, with the differentiated neighbor types, a free-rider can only get limited benefit even if it keeps jumping around. With the extreme setup $d_s = 0$, no matter how many receptors the free-rider actively locate, it will not be served for free.

Algorithm 1: Receiver side scheduling in one round

```
input : $\mathcal{K}_n$: set of neighbors of peer $n$;
        $\mathcal{C}_n$: set of LCs to be scheduled;
        $l_n$: number of layers for regular requests;
        $L$: total number of layers;
        $layer\_idx[i]$: the layer index of LC $i$;
output: LC schedule;
randomize $\mathcal{C}_n$;
forall $i \in \mathcal{C}_n$ and $layer\_idx[i] \leq l_n$ do
    forall $k \in \mathcal{K}_n$ do
        if LC $i$ is available at neighbor $k$ then
            insert $k$ to $supplier\_set$;
        end
    end
    randomly select a neighbor $k^* \in supplier\_set$ to
    serve LC $i$;
end
for $l \leftarrow l_n + 1$ to $L$ do
    forall $i \in \mathcal{C}_n$ and $layer\_idx[i] = l$ do
        forall $k \in \mathcal{K}_n$ do
            if LC $i$ is available at neighbor $k$ then
                insert $k$ to $supplier\_set$;
            end
        end
        randomly select a neighbor $k^* \in supplier\_set$
        to serve LC $i$;
    end
end
```

IV. Implementation

Following the design philosophy described in the previous section, we have created an implementation of LayerP2P, including tracker, seed, clients, and layered video encoder and decoder.

We first describe the video encoding and LC segmentation modules. Although LayerP2P can employ any layered video coding scheme, we use H.264 temporal scalable video in our current implementation. This is mainly because H.264 temporal scalable video does not lose any video coding efficiency compared with H.264 single-layer video. Furthermore, a well-designed, real-time decoder, FFmpeg[8], which can decode H.264 temporal scalable video, is available in the public domain. Figure 4 illustrates the architecture of temporal scalable video with Bs and B frames in H.264. Our current implementation encodes video into three layers, with layer 1 containing all I and P frames, layer 2 containing all Bs frames, and layer 3 containing all B frames. Note that when layer 3 LCs are not received, they do not affect the decoding of other frames in any layer; but when layer 1 LCs in a GOP are lost, they affect decoding of all subsequent frames in this GOP.

Our current prototype uses the H.264 encoder JM11 [15]. To achieve consistent video quality, we disable the rate control in JM11 and fix the quantization parameters (QPs) for I, P, B frames in encoding. The resulting video has variable bit rate. Recall that each of three layers is a sequence of frames. For each layer, the source segments it into slices. Each slice

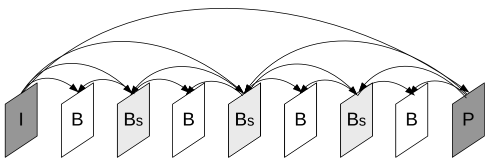

Fig. 4. Coding architecture of temporal scalable video in H.264.

is set to be less than 1250 bytes, so that it can fit into a RTP/UDP/IP packet. A frame requiring more than 1250 bytes is split into multiple slices. Note that a slice never contains data from multiple frames, so that its loss does not affect more than one frame. Each slice is inserted into one RTP packet, which is treated as one LC. At the receiver we use MPlayer [8], which uses FFmpeg as the decoder core, employing its default scheme for error concealment (when LCs are lost).

The LayerP2P engine, implemented in C++, is the core software piece. It obtains peer lists from the tracker, chooses neighbors for overlay formation, exchanges LC-maps with neighbors, and performs both supply and receiver side scheduling of LCs, as described in the previous section. LayerP2P uses a connectionless design, where both the signaling messages (e.g., buffer maps, neighbor management messages) and video chunks are delivered over UDP. Note that all video chunks are delivered based on explicit LC requests. The client allows the user to freely set its maximum upload bandwidth. The system playback lag, i.e., the lag between a live event being encoded and sent at the source and that being played at the peers, is set to 30 seconds. The round time, T, is set to one second for sending buffer maps and re-scheduling LCs. The buffer maps and request notifications are piggybacked in the video chunks. $N_{\min}$ and $N_{\max}$ are set to 10 and 20, respectively. A peer adapts its neighbors every 30 seconds.

V. PlanetLab Experiment

We validate the LayerP2P philosophy by conducting a large-scale experiment over PlanetLab. We use a node located at UC Berkeley as the tracker, a node located at MIT as the seed, and another 100 nodes located all over the world as peers. Several applications are typically running simultaneously over any given PlanetLab node, making PlanetLab an even more dynamic network environment than the real Internet.

In the experiment, we apply H.264 encoder JM11 [15] on the video sequence "ICE" in 4CIF (704x576) resolution with a frame rate of 30 frames/sec. The output bit rate is varying, with the average total bitrate being about 620 kbps. The average bitrates of layer 1, 2, 3 are 290 kbps, 230 kbps, and 100 kbps, respectively. At the video source, we loop the 10-second ICE video sequence to construct a long video sequence. We encode the video in "IBBsBBsBBsBP" structure as shown in Fig. 4. In our settings, one GOP includes 64 frames.

We mimic three types of peers by limiting the upload rates of the PlanetLab peers: (i) institutional peers; (ii) residential peers; and (iii) free-riders. We set the upload bandwidth of the seed to 2 Mbps. We consider two scenarios: (i) an underloaded system without the presence of free-riding and (ii) an overloaded system with the presence of free-riding.

We assume that in the underloaded scenario, peers are willing to contribute their upload bandwidth, while in the overloaded scenario, peers are reluctant to contribute. Detailed setups are shown in Table I. For example, in the underloaded scenario, 40% of peers are residential peers with an upload bandwidth contribution of 400 kbps, 60% of peers are institutional peers with an upload bandwidth contribution of 1 Mbps, and there is no free-rider.

We simulate a flash crowd in our experiments, where all peers connect in the system during the first 120 seconds with a uniform distribution. After that, all peers stay in the system for 1200 seconds. We run each scenario five times to obtain the statistics. In our trace-driven simulation in Section VI, we consider peer churn, with real peer upload bandwidth distributions and peers dynamics.

A. Single-Layer Systems

For comparison purposes, we also developed two single-layer video systems:

- **Single-Layer:** Single-Layer treats all peers equally, without taking into account their upload contributions. As a supplier, a peer serves its receivers in a round robin fashion. The video is divided into video chunks with each chunk including one video slice. Single-Layer treats all video chunks equally. Receivers request the missing video chunks with the random scheduling algorithm. In Single-Layer, we use the same hierarchical B structure as shown in Figure 4 to encode the video with the same parameters. Note that the hierarchical B structure provides a higher video coding efficiency than H.264 video coding not using hierarchical B structure. This system is considered as a representative of most existing P2P live streaming systems.
- **Single-Incent:** Single-Incent takes into account the upload contributions of peers. It applies the tit-for-tat strategy described in Section III-D when a peer acts as a supplier. However, unlike LayerP2P, Single-Incent uses single-layer video instead of layered video. Like Single-Layer, Single-Incent treats all video chunks equally with the random scheduling algorithm.

Careful attention is placed on making the comparisons as fair as possible, with the same peer upload bandwidth distribution, P2P engine configurations, and video coding setups.

B. Performance Metrics

Three metrics are adopted for performance evaluation:

- **Playback rate (R):** Playback rate is the received bitrate of a particular peer used for decoding the video. It counts all received LCs from all layers that arrived before their playback deadlines.
- **Received chunk ratio (α):** Playback rate can largely determine the received video quality. However, with layered video, the video quality is also affected by the specific LCs received. We investigate the percentage of received chunks for different layers.
- **Average PSNR (Q):** PSNR is widely adopted to represent the decoded video quality. We calculate the PSNR for each decoded frame and average PSNR over different sets of frames. For temporal scalable video, it is less meaningful to only show the average PSNR over all frames, since some frames from higher layers are automatically skipped and reconstructed by error concealment from lower frames. To have a full understanding of the video quality, we show the average PSNRs of the frames from different layers separately. We let Q_1 represent the average PSNR over all I and P frames, Q_2 represent that over all Bs frames, and Q_3 represent that over all B frames. We also show the average PSNR Q over all video frames.

C. Experimental Results

1) Underloaded Scenario without the Presence of Free-Riding: We begin by considering an underloaded scenario, where the system bandwidth supply is higher than the system bandwidth demand. We further assume all peers are cooperative and there are no free-riders. This is considered as a "healthy" situation in P2P live streaming. Part of our design philosophy is to provide maximal video quality to all peers when the system is in this state. We investigate whether LayerP2P achieves this goal. We use the first set of bandwidth distributions in Table I. In this case, the average upload bandwidth of the system is 760 kbps (which exceeds the average video rate 620 kbps). The resource index, i.e., the ratio of the average upload bandwidth over the full video rate, is 1.23. Note that some PlanetLab nodes may not realize the assigned upload bandwidth, so that the actual resource index may be smaller.

Figure 5 shows the playback rates at two randomly chosen peers, one with an upload bandwidth of 1 Mbps, another with an upload bandwidth of 400 kbps, in LayerP2P. We observe that each peer receives a high video rate (close to the full video rate). The rate fluctuation is due to the variation of the actual video rate. Both peers reach a stable state within 100 seconds. Once a peer reaches its stable state, the video quality is generally smooth, without significant variation. Figure 5 also shows the playback rate for a randomly selected peer in Single-Layer and those for two peers in Single-Incent. Similarly, the single-layer systems can also perform well under such an underloaded scenario.

Figure 6 shows the CDFs of the playback rates across all video sessions in LayerP2P, Single-Layer, and Single-Incent. We observe that for all of the three systems and both types of peers [1], the peers receive a high playback rate. LayerP2P can efficiently use the surplus bandwidth of the high-upload-bandwidth peers, and help the low-upload-bandwidth peers to obtain good video quality. Note that to have an underloaded system, the system needs altruistic behavior from rich-bandwidth peers, with these peers being willing to upload at rates higher than the video rate.

We examine the received chunk ratio for all of the three systems. As shown in Figure 7, for both types of peers, the peers enjoy a high received chunk ratio for all three layers.

[1] Single-Layer does not differentiate different types of peers, hence we only plot one curve in the figure.

TABLE I
PEER UPLOAD BANDWIDTH DISTRIBUTION IN PLANETLAB EXPERIMENTS (KBPS)

Peers		Free-rider	Residential	Institutional
Underloaded&No Free-Riding	Distribution (%)	-	40%	60%
	Actual upload rate	-	400	1000
Overloaded&Free-Riding	Distribution (%)	10%	30%	60%
	Actual upload rate	0	300	700

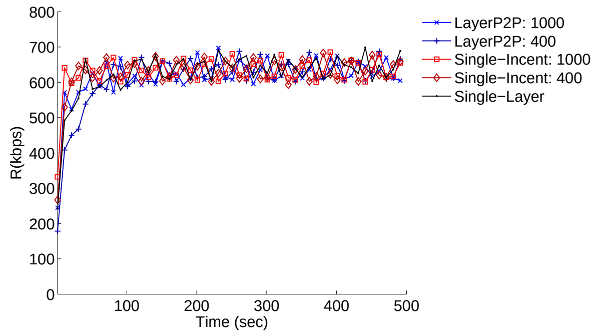

Fig. 5. Behavior of typical peers in an underloaded system in Planetlab Experiment.

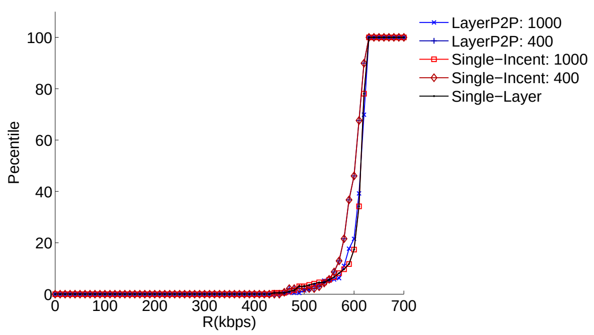

Fig. 6. Cumulative distribution of average download rate in an underloaded system in Planetlab Experiment.

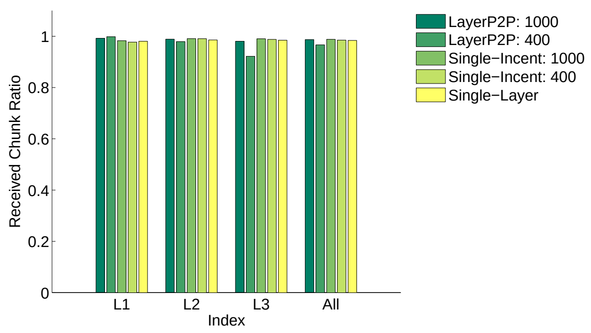

Fig. 7. Received chunk ratio for different layers in an underloaded system in Planetlab Experiment.

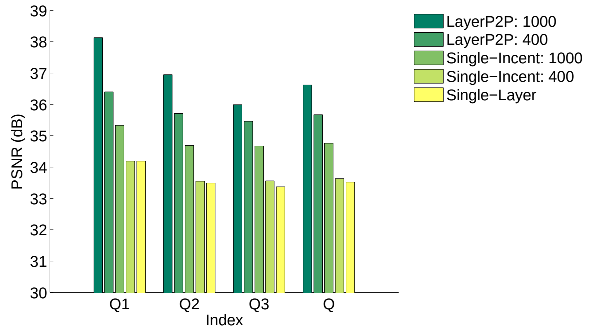

Fig. 8. Average PSNR for different layers in an underloaded system in Planetlab Experiment.

LayerP2P provides more protection to lower layers, so that the received chunk ratios for lower layers are normally higher than those for higher layers. Unlike LayerP2P, Single-Layer and Single-Incent treat all three layers equally, so that the received chunk ratios for all three layers are almost the same. We also indicate the average received chunk ratio over all three layers. We can see that the overall received chunk ratio is similar for all three systems and both types of peers.

We use PSNRs (Q_1, Q_2, Q_3, Q) to examine the received video quality. Figure 8 shows the average PSNRs for different types of peers in all three systems. We can observe that the institutional peers in LayerP2P receive the best video quality with the highest PSNRs (individual PSNRs $Q1$, $Q2$ and $Q3$ and the overall PSNR Q). The overall PSNR is as high as 36.62 dB. The video quality of the residential peers in LayerP2P is slightly lower, with an overall PSNR being 35.67 dB. Due to the unequal protection to different layers, both types of peers in LayerP2P receive a higher video quality than those in the single-layer video systems. The gaps between LayerP2P and the single-layer systems for $Q1$, $Q2$, and Q are more than 2 dB. This is because with LayerP2P, the LCs from more important layers are more likely to be received.

2) Overloaded Scenario with Free-Riding: We now consider a more challenging scenario, where the average upload bandwidth of the system is lower than the full video rate. In the highly dynamic P2P environment, it is possible that the system enters (or permanently lives in) such bandwidth deficient situations, especially when video rate is high. It is desirable for the system to adapt to the overloaded situation, with each peer receiving video quality that is commensurate with its contribution. Additionally, we consider free-riding, which can potentially bring a P2P live streaming system to its knees. A P2P live streaming system should discourage free-riding, by providing poor video quality for free-riders. In this experiment, we use the second set of bandwidth distributions in Table I. The average upload bandwidth is 520 kbps, leading to a resource index of 0.82.

Figure 9 shows the typical received rate of the three types of peers in an overloaded scenario. We observe that in LayerP2P, an institutional peer has a similar received rate as with the underloaded system. It can achieve a high video rate (close to the full video rate) soon after the video session starts. Unlike the underloaded system, the residential peer receives a lower video rate which is commensurate to its upload contribution. The free-rider receives a very low rate with

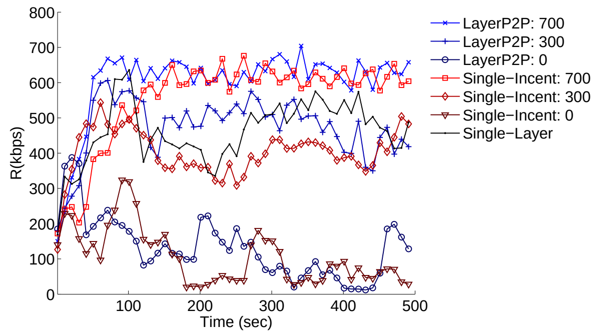

Fig. 9. Behavior of typical peers in an overloaded scenario in PlanetLab Experiment.

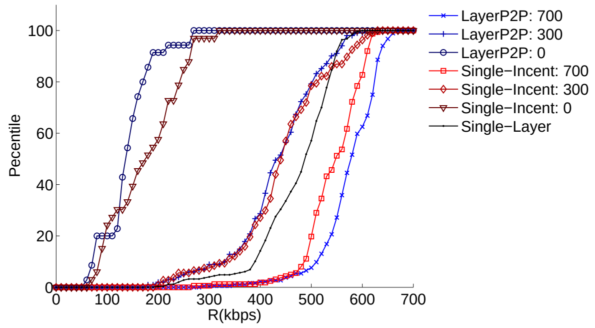

Fig. 10. Cumulative distribution of average download rate in an overloaded scenario.

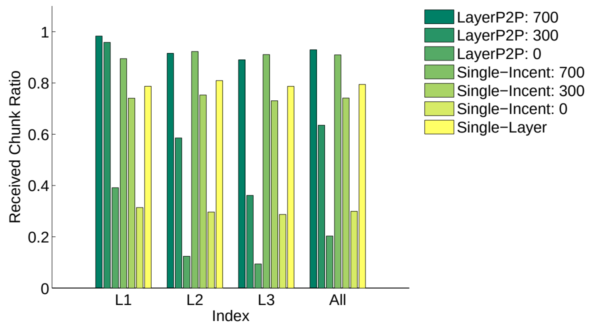

Fig. 11. Received chunk ratio for different layers in an overloaded scenario.

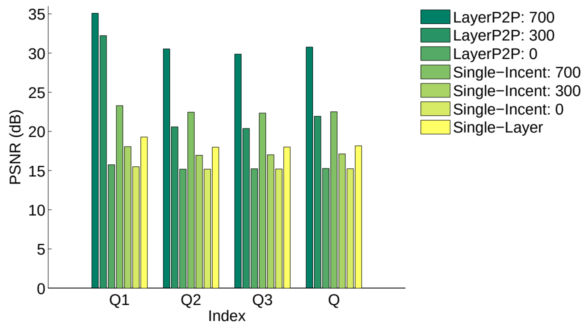

Fig. 12. Average PSNR for different layers in an overloaded scenario in PlanetLab Experiment.

significant fluctuation. At the beginning of the video session, the free-rider is treated as a newcomer and receives a medium download rate for free; after the initial period, its download rate drops dramatically below the rate of the first layer. In such an overloaded system where the system supply is below the system demand, not all peers can receive the full video rate. As shown in the figure, the peer in Single-Layer can only receive a partial video rate. Note that in Single-Layer, all peers receive similar rates, regardless their upload contribution. Single-Incent can also provide differentiated services. Similar to LayerP2P, different types of peers receive a different video rate, due to the tit-for-tat strategy.

Figure 10 further demonstrates the differentiated services provided by LayerP2P. The institutional peers receive the highest playback rate. More than 70% of institutional peers receive a rate higher than 550 kbps. The residential peers receive a moderate playback rate. More than 70% of residential peers receive a rate higher than 400 kbps. Importantly, the free-riders receive a low playback rate. More than 90% of free-riders receive a rate lower than 200 kbps. Similarly, Single-Incent also leads to a differentiated received video rate. Unlike LayerP2P and Single-Incent, Single-Layer does not provide differentiated services to different types of peers.

Figure 11 shows the received chunk ratios for different layers. In LayerP2P, the institutional peers can receive a high received chunk ratio for all three layers, which indicates a good reconstructed video quality. The residential peers have lower received chunk ratios. But due to the prioritized random scheduling algorithm, lower layers get more protection and have higher received ratios. For example, the received chunk ratio of the lowest layer is high ($> 95.8\%$). It largely guarantees the basic video quality. For free-riders, the received chunk ratios for all three layers are very low. Single-Layer does not differentiate between different video layers, so that all three layers have a similar received chunk ratio (about 80%). Similarly, Single-Incent does not differentiate between different video layers.

We now examine received video quality using PSNR. As shown in Figure 12, in LayerP2P, the institutional peers receive a significantly higher video quality (30.8 dB) than other types of peers. Given such a low resource index (0.82), LayerP2P can provide a reasonably good video quality to the peers that have a higher upload rate (700 kbps) than the full video rate (620 kbps). Consistent with the received chunk ratio, the residential peers receive a lower overall video quality. However, the residential peers have a good reconstructed quality for frames from layer 1 with $Q1$=32.3 dB. This indicates that these peers can receive the basic video quality. In LayerP2P, the free-riders receive a very poor video quality (15.8 dB), which is unacceptable for video service. This virtually prevents free-riding. In Single-Layer, all peers receive a significantly degraded overall video quality ($Q = 18.2$ dB). The reconstructed video qualities for the individual layers are also low. This indicates that under such an overload scenario, most peers in Single-Layer cannot receive an acceptable video quality. Single-Incent leads to a similar download rate for different types of peers to LayerP2P, however, its reconstructed video quality is significantly lower than that of LayerP2P. With single-layer video (without unequal protection to different layers in our experiment), even though the institution peers can receive relatively high received chunk ratios (about 90%) for all three layers, the reconstructed video quality is low

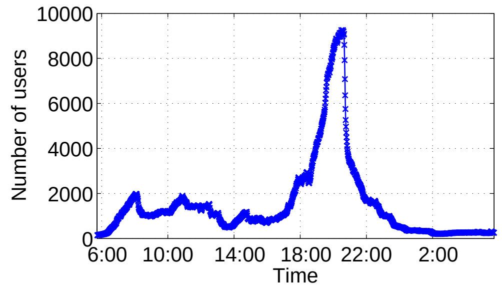

Fig. 13. Evolution of number of users viewing a TV channel in the PPLive network.

($Q = 22.51$ dB) due to error propagation. The residential peers in Single-Incent receive an even worse video quality. In such an overload system, LayerP2P uses the limited bandwidth resource more efficiently than Single-Layer and Single-Incent in terms of decoded video quality. Layered coding allows a receiver to reconstruct the basic video quality with a partial video rate. Therefore, peers in LayerP2P are all self-sustained – a peer can at least receive a video quality commensurate with its upload contribution. With single-layer video, the peers with low upload bandwidth highly depend on the altruistic behavior of the peers with high upload bandwidth. Note that in Single-Incent, the residential peers are not self-sustained – if a peer cannot trade a full video rate, the decoded video quality will be significantly degraded and unacceptable.

In summary, deploying our implementation in PlanetLab, we have learned that in an underloaded scenario (that is, with altruistic high-capacity peers), the LayerP2P system provides maximal quality to all peers. In an overloaded scenario, where it is not possible for all peers to get the full video quality, the peers that upload at higher rates receive higher video quality, while the peers that upload at lower rates receive lower but still acceptable video quality. And the system discourages free-riding by providing the free-riders a poor video quality (when free-riding brings the system into an overloaded situation).

VI. Trace-Driven Simulation

Our studies on PlanetLab validates LayerP2P in a real Internet environment. However, PlanetLab experiments can only involve hundreds of nodes (in our case, 100+). To investigate the performance of LayerP2P in a large scale network with real peer dynamics, we conduct a trace-driven simulation by using the traces for peer dynamics from a real-world P2P live streaming system.

A. Simulation Setup

We developed a chunk-level discrete-event simulator in C++. With this simulator, all actions are performed at the chunk-level. In our simulations, we assume that the end-to-end bandwidth bottleneck is at the access links but not in the Internet core. The simulator simulates all behaviors in LayerP2P for LC (video packet) propagation. For example, each peer maintains its received LCs in a virtual buffer, and exchanges these LCs with its neighbors. We assume the transmission delay of LCs (video packets) is determined by the upload bandwidth of the supplier rather than the core network. We do not simulate the transmission delay for the signaling messages and assume the signaling messages can be received immediately by a receiver. Such an abstraction speeds up the simulation and we believe it can still give us an accurate evaluation of the system.

Peer dynamics are simulated by traces collected from PPLive [2], a real-world P2P live streaming system. The traces record the arrival and departure times of the users for different channels. We select the trace of a popular Chinese TV channel, CCTV3, to drive our simulations. This one-day trace had totally more than 100,000 video sessions during the period. Figure 13 shows the evolution of number of users viewing this channel. This trace covers a variety of typical scenarios in P2P live video networks, such as small systems (less than 200 concurrent users), large systems (more than 9,000 concurrent users), short video sessions (shorter than one minute), long video sessions (longer than 16 hours), and flash crowds.

To come up with a realistic upload bandwidth distributions, we combine the measurement studies in [16] and [17]. The overall distribution of residential peers and Ethernet peers is obtained from [16], while the detailed bandwidth distribution of residential peers is obtained from [17]. (We exclude modem peers and ISDN peers due to their very low upload and download capacities.) Because peers may not be willing to contribute their entire upload bandwidth, in our simulations we assume that the peers with high upload bandwidth only contribute portions of their upload bandwidth, which are indicated in Table II. For example, in contribution set I, the 1500 kbps peers contribute 1000 kbps upload bandwidth. In our experiments, the upload contribution of the various peers are assigned according to the distribution of Table II.

In the simulations, we use the same peer/server configurations presented in the PlanetLab experiments. We repeat the simulations under the underloaded and overloaded scenarios discussed in Section V. We do not use real encoded video in this case, and simply assume that the video is encoded into three layers, with each layer having a constant rate of 200 kbps and each LC being 1250 bytes. We use playback rate R and received chunk ratio α as the performance metrics. Since no real encoded video is applied, we will not provide PSNR in this case.

B. Simulation Results

We first consider an underloaded scenario. We assume that all peers are willing to contribute their upload bandwidth, as indicated in the “underloaded” setup in Table II. In this case, the resource index is about 1.26, which indicates a rich resource system. Figure 14 shows the CDF of the average playback rate for Single-Layer, Single-Incent, and LayerP2P in our simulations. We select two types of peers from ten types of peers to present, the peers with 1024 kbps upload bandwidth and those with 448 kbps upload bandwidth. For all of the three systems, almost all selected peers can receive the full video rate. Figure 15 shows that almost all video chunks have been successfully received.

For an overloaded scenario, as indicated in the “overloaded” setup in Table II, the peers with low upload bandwidth are

TABLE II
PEER UPLOAD BANDWIDTH DISTRIBUTION (KBPS)

Total upload bandwidth (kbps)	256	320	384	448	512	640	768	1024	1500	> 3000
Distribution (%)	10.0	14.3	8.6	12.5	2.2	1.4	6.6	28.1	1.4	14.9
Actual upload rate in underloaded scenario	240	320	360	400	480	600	720	920	1000	1800
Actual upload rate in overloaded scenario	0	0	0	200	280	400	600	720	960	1000

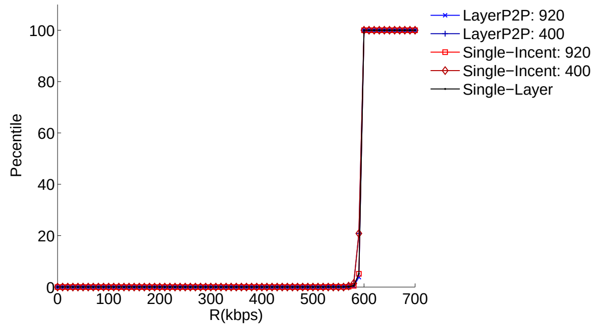

Fig. 14. CDF of average playback rate in underloaded scenario in trace-driven simulation.

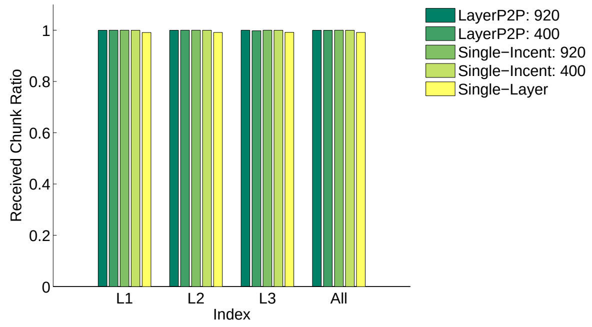

Fig. 15. Received chunk ratio for different layers in underloaded scenario in simulation.

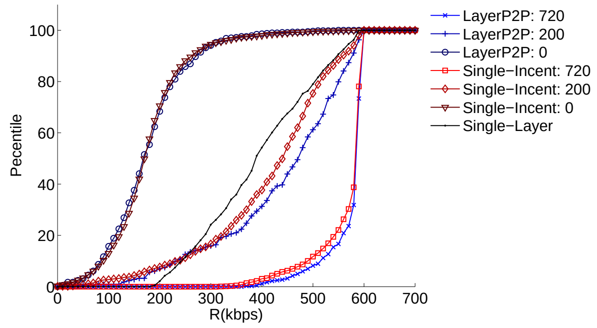

Fig. 16. CDF of average playback rate in an overloaded scenario in trace-driven simulation.

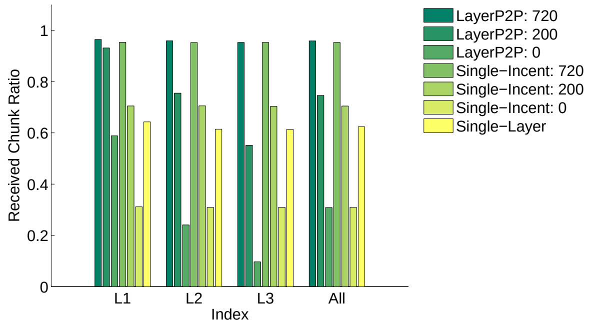

Fig. 17. Received chunk ratio for different layers in overloaded scenario in simulation.

chosen to be free-riders who do not contribute any upload bandwidth. Compared with the underloaded system, the high-upload-bandwidth peers also contribute less upload bandwidth. In this case, the resource index reduces to 0.73. Similar observations can be made from Figure 16 and Figure 17 as those in the PlanetLab experiments. Single-Layer degrades more dramatically compared with LayerP2P. Peers can only receive 60% of the video chunks. In LayerP2P, the peers with high upload bandwidth enjoy high received chunk ratios ($> 95\%$) for all layers; the peers with low upload bandwidth obtain unequal received chunk ratios for different layers, but a high received chunk ratio ($> 93\%$) for the base layer; the free-riders receive low received chunk ratios for all layers. The simulations also show that Single-Incent provides differentiated download rate for different types of peers, but it ignores different importance of different layers.

Note that the number of peers ranges from less than 200 to more than 9,000 in the simulation. We observed that in both the small system (with less than 200 peers) and the large system (with more than 9,000 peers), LayerP2P consistently had excellent performance. This demonstrates the scalability of LayerP2P: the enforced trading relationships among peers do not reduce the system's scalability. Furthermore, we observed that LayerP2P handles peer churn well (e.g., during the period 18:00 to 22:00), demonstrating that LayerP2P is robust to peer dynamics.

VII. Related Work

Over the past few years, there has been a number of proposals for live P2P video in the research community [18], [19], [20], [21], [1]. These papers show the feasibility of using P2P for delivering live video content to a large number of users. However, none of these research considers adapting the system resources along with providing built-in robustness to uncooperative peers.

These proposals have an implicit assumption that the resource index in the system is larger than 1. Layered video has been proposed in P2P streaming systems to address the download heterogeneity of peers [22]. Unlike this previous work, we apply layered video to address a much broader set of problems, including incentives for redistribution, adaptation for supply/demand, and video quality improvement. We have developed a working prototype, deployed it in PlanetLab, and demonstrated the feasibility of the approach.

General theories for incentives in P2P are developed in [23], [24]. To date little work has been done in providing incentives in P2P video live streaming. Mol *et al.* propose an MDC-based multiple-tree scheme that employs tit-for-tat incentives [25]. Each description is distributed over a separate tree, and peers belonging to different trees exchange descriptions with each other. This approach is based on MDC (which is less efficient compared with layered video [26]), cannot be easily adapted to layered video or single-layer video, and restricts a peer to trade only the description corresponding to the tree to which it belongs. Pianese *et al.* propose a chunk-based mesh-pull scheme with single-layer video [27]. The scheme applies a combination of tit-for-tat and donation strategies to provide incentives. However, this scheme is limited to single-layer video. We [28] have recently proposed a tit-for-tat scheme for MDC and layered video using "substream trading". In that scheme, substreams, rather than chunks, are traded among peers. We use simulations to evaluate the scheme. In comparison, this paper presents a chunk-based scheme (i) which is significantly less complex; and (ii) whose viability is demonstrated with a working prototype that employs actual layer-encoded video.

The basic idea of LayerP2P was presented in our earlier six-page workshop paper [29]; the workshop paper does not include an implementation or experimental results. The current paper significantly extends and refines LayerP2P, and demonstrates the viability of the approach with extensive PlanetLab experiments and trace-driven simulations. An important new contribution is our prototype, a complete implementation of LayerP2P, including layered codecs and our incentive-centric P2P engine.

VIII. Conclusion

Although there are several successful commercial deployments of live P2P streaming systems, the current designs (i) lack incentives for users to contribute bandwidth resources, (ii) lack adaptation to aggregate bandwidth availability, and (iii) exhibit poor video quality when bandwidth availability falls below bandwidth supply. In this paper, we proposed, prototyped, deployed and validated *LayerP2P*, a P2P live streaming system that addresses all three of these problems. LayerP2P combines layered video, mesh P2P distribution, and a tit-for-tat-like algorithm, in a manner such that a peer contributing more upload bandwidth receives more layers and consequently better video quality. This provides built-in incentives for peers to re-distribute, and discourages free-riders. Moreover, our chunk requesting and scheduling schemes give higher priority to more important layers, so that the peers can receive the base layer with a high probability.

We implemented LayerP2P (including seeds, clients, trackers, and layered codecs), deployed the prototype in PlanetLab, and performed extensive experiments. We also examined a wide range of scenarios using trace-driven simulations. The results in Sections V and VI showed that LayerP2P has high efficiency, provides differentiated service, adapts to bandwidth deficient scenarios, and provides protection against free-riders. Compared to two P2P live streaming systems with single-layer video, LayerP2P provides significantly improved video quality for cooperative peers.

We also believe that LayerP2P can serve as a framework for an open design for P2P live streaming systems. With an open P2P design, it becomes necessary to incorporate an incentive mechanism to encourage peers to contribute upload bandwidth. Without such an incentive in an open protocol, clients can easily be written to free-riders or be configured to upload at low rates. LayerP2P provides the requisite incentives for an open design.

References

[1] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "DONet: A data-driven overlay network for efficient live media streaming," in *IEEE INFOCOM*, March 2005.

[2] http://www.pplive.com/.

[3] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P VoD system," in *ACM SIGCOMM*, August 2008.

[4] http://www.ppstream.com/.

[5] http://www.uusee.com/.

[6] ISO/IEC, "ISO/IEC JTC1/SC29/WG11. N9560," January 2008.

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

[8] http://ffmpeg.mplayerhq.hu/.

[9] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE TCSVT*, vol. 17, no. 9, September 2007.

[10] B. Cohen, "Incentives build robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[11] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang, "Exploiting BitTtorrent for fun (but not profit)," in *IPTPS*, February 2006.

[12] M. Zhang, Q. Zhang, and S.-Q. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?" *IEEE JSAC*, vol. 25, no. 9, December 2007.

[13] T. Bonald, L. Massoulie, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: optimal performance trade-offs," in *ACM SIGMETRICS*, June 2008.

[14] S. Agarwal, M. Laifenfeld, A. Trachtenberg, and M. Alanyali, "Fast data access over asymmetric channels using fair and secure bandwidth sharing," in *IEEE ICDCS*, July 2006.

[15] Alexis Michael Tourapis, Karsten Sühring, and Gary Sullivan, "Revised H.264/MPEG-4 AVC reference software manual," *Joint Video Team, Doc. JVT-Q042*, October 2005.

[16] C. Huang, J. Li, and K. W. Ross, "Can Internet VoD be profitable?" in *ACM SIGCOMM*, August 2007.

[17] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *ACM IMC*, October 2007.

[18] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS*, June 2000.

[19] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *ACM NOSSDAV*, May 2003.

[20] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment," in *IPTPS*, February 2003.

[21] V. Venkataraman, P. Francis, and J. Calandrino, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *IEEE ICNP*, November 2006.

[22] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming," in *ACM NOSSDAV*, June 2003.

[23] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," in *EC*, 2004.

[24] B. Q. Zhao, J. C. S. Lui, and D.-M. Chiu, "Mathematical modeling of incentive policies in P2P systems," in *NetEcon*, 2008.

[25] J. D. Mol, D. H. P. Epema, and H. J. Sips, "The Orchard algorithm: Building multicast trees for P2P video multicasting without free-riding," *IEEE Trans. on Multimedia*, vol. 9, no. 8, December 2007.

[26] F. H. Fitzek, B. Can, R. Prasad, and M. Katz, "Overhead and quality measurements for multiple description coding for video services," in *WPMC*, September 2004.

[27] F. Pianese, D. Perino, J. Keller, and E. W. Biersack, "PULSE: an adaptive, incentive-based, unstructured P2P live streaming system," *IEEE Trans. on Multimedia*, vol. 9, no. 8, December 2007.

[28] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "Substream trading: Towards an open P2P live streaming system," in *IEEE ICNP*, October 2008.

[29] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Using layered video to provide incentives in P2P streaming," in *ACM SIGCOMM P2P-TV*, August 2007.

Zhengye Liu received the B.S. and M.S. degrees in Electronic Engineering from Tsinghua University, Beijing, China, in 2000 and 2003, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Polytechnic Institute of New York University in 2009. He has been a postdoctoral researcher in the Department of Electrical and Computer Engineering in Polytechnic Institute of New York University since March 2009. He interned in the Corporate Research, Thomson Inc., NJ in 2005 and 2007. His research interests include networked multimedia, video communications, peer-to-peer networking, and wireless networking.

Yanming Shen is an Associate Professor in the Computer Science and Engineering Department at Dalian University of Technology, China. He received the Ph.D. degree from Department of Electrical and Computer Engineering at the Polytechnic University (now Polytechnic Institute of NYU) and his B.S degree in Automation from Tsinghua University, Beijing, P.R.China in 2000. He was a summer intern with Avaya Labs in 2006, conducting research on IP telephony. His general research interests include packet switch design, peer-to-peer video streaming, and algorithm design, analysis and optimization.

Keith W. Ross (S'82-M'86-SM'90-F'06) joined Polytechnic University as the Leonard J. Shustek Chair Professor in Computer Science in January 2003. He has been Department Head since September 2008. Before joining Polytechnic, he was a professor for five years in the Multimedia Communications Department at Eurecom Institute in Sophia Antipolis, France. From 1985 through 1997, he was a professor at the University of Pennsylvania. He received a B.S.E.E from Tufts University, a M.S.E.E. from Columbia University, and a Ph.D. in Computer and Control Engineering from The University of Michigan. He has worked in peer-to-peer networking, Internet measurement, video streaming, Web caching, multi-service loss networks, content distribution networks, network security, voice over IP, optimization, queuing theory, and Markov decision processes. He is an IEEE Fellow, recipient of the Infocom 2009 Best Paper Award and recipient of Best Paper in Multimedia Communications 2006-2007 (awarded by IEEE Communications Society). He has served on numerous journal editorial boards and conference program committees, and was the PC co-chair for ACM Multimedia 2002, ACM CoNext 2008, and IPTPS 2009. He has served as an advisor to the Federal Trade Commission on P2P file sharing. Professor Ross is co-author (with James F. Kurose) of the popular textbook, Computer Networking: A Top-Down Approach Featuring the Internet, published by Addison-Wesley (first edition in 2000, fifth edition 2009. Professor Ross is also the author of the research monograph, Multiservice Loss Models for Broadband Communication Networks, published by Springer in 1995.

Shivendra S. Panwar (S'82-M'85-SM'00) received a B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1981, and M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, in 1983 and 1986, respectively. He joined the Department of Electrical Engineering at the Polytechnic Institute of New York, Brooklyn (now the Polytechnic Institute of NYU), where he is a professor in the Electrical and Computer Engineering Department. Currently, he is the director of the New York State Center for Advanced Technology in Telecommunications (CATT). His research interests include the performance analysis and design of networks. Current work includes video systems over peer-to-peer networks, switch performance, and wireless networks. He spent the summer of 1987 as a visiting scientist at the IBM T. J. Watson Research Center, Yorktown Heights, New York, and was a consultant to AT&T Bell Laboratories, Holmdel, New Jersey. He has served as the secretary of the Technical Affairs Council of the IEEE Communications Society (1992-1993) and is a member of the Technical Committee on Computer Communications. He is a co-editor of two books: Network Management and Control, vol. 2, and Multimedia Communications and Video Coding (Plenum, 1994 and 1996, respectively) and co-author of TCP/IP Essentials: A Lab-Based Approach (Cambridge University Press, 2004). He is a co-recipient of the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems.

Yao Wang (M'90-SM'98-F'04) received the B.S. and M.S. degrees in Electronic Engineering from Tsinghua University, Beijing, China, in 1983 and 1985, respectively, and the Ph.D. degree in Electrical and Computer Engineering from University of California at Santa Barbara in 1990. Since 1990, she has been with the Electrical and Computer Engineering faculty of Polytechnic University, Brooklyn, NY (now Polytechnic Institute of New York University). Her research interests include video coding and networked video applications, medical imaging and pattern recognition. She is the leading author of the textbook Video Processing and Communications. She has served as an Associate Editor for IEEE Transactions on Multimedia and IEEE Transactions on Circuits and Systems for Video Technology. Dr. Wang received New York City Mayor's Award for Excellence in Science and Technology in the Young Investigator Category in year 2000. She was elected Fellow of the IEEE in 2004 for contributions to video processing and communications. She is a co-winner of the IEEE Communications Society Leonard G. Abraham Prize Paper Award in the Field of Communications Systems in 2004. She received the Overseas Outstanding Young Investigator Award from the Natural Science Foundation of China in 2005 and was named Yangtze River Lecture Scholar by the Ministry of Education of China in 2007.