Substream Trading: Towards an Open P2P Live Streaming System

Zhengye Liu†, Yanming Shen†, Keith W. Ross‡, Shivendra S. Panwar†, and Yao Wang†
Dept. of Electrical and Computer Engineering†
Dept. of Computer and Information Science‡
Polytechnic University, NY 11201

ABSTRACT

We consider the design of an open P2P live-video streaming system. When designing a live video system that is both open and P2P, the system must include mechanisms that incentivize peers to contribute upload capacity. We advocate an incentive principle for live P2P streaming: a peer's video quality is commensurate with its upload rate.

We propose Substream Trading, a new P2P streaming design which not only enables differentiated video quality commensurate with a peer's upload contribution but can also accommodate different video coding schemes, including single-layer coding, layer coding, and multiple description coding. Extensive trace-driven simulations show that substream trading has high efficiency, provides differentiated service, low start-up latency, synergies among peers with different Internet access rates, and protection against free-riders.

1. INTRODUCTION

BitTorrent is a remarkably popular file-distribution technology, with millions of users sharing content in hundreds of thousands of torrents on a daily basis. Fundamental to BitTorrent's success is its *openness* – the BitTorrent protocol is published in the Internet, and the source code of the baseline implementation is made widely available. This openness has enabled developers to create over 50 independent BitTorrent client implementations [1], dozens of independent tracker implementations [2], and a multitude of torrent search sites. The openness of the protocol has further fostered open discussions in both the online developer and research communities, leading to further performance and security improvements. It has also fostered innovations in the broader BitTorrent ecosystem, including recent deployments of distributed trackers, using DHTs and gossiping, in many popular BitTorrent clients.

A second key element in BitTorrent's success is, of course, its P2P design. Since BitTorrent peers assist in file distribution, a file can be distributed to an unlimited number of peers with modest initial seeding capacity.

But 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-ride (that is download without uploading) or be configured to upload at low rates. Bram Cohen, the inventor of the original BitTorrent system, recognized the need of building into the system a simple, but effective incentive mechanism [5]. Fundamentally, 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 [5], 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 [12, 14, 19, 23], the algorithm has nevertheless been very successful in practice.) 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 high download times.

In this paper, inspired by BitTorrent's open and P2P philosophies, we consider the design of an *open P2P live-video streaming system*. Ideally, such a design would lead to an open protocol and numerous independent client, seed, and tracker implementations. Such a live P2P streaming system would also allow arbitrary users to seed live video channels – including live user-generated content – in much the same way that BitTorrent allows arbitrary users to seed files. Eventually, we expect much of the live content to emanate from handheld wireless devices, and may include such diverse sources as professors' lectures, Little League baseball games, and political demonstrations.

Recently, several P2P live video systems have been successfully deployed. They have reported phenomenal success on their Web sites, claiming tens of thousands of simultaneous users in a single channel, with stream rates between 300 kbps to 1 Mbps. These systems include CoolStreaming [32], PPLive [20, 9], PPStream [21], UUSee [25] and many more. The success of these systems shows the potential of broadcasting live video over P2P networks. However, all of these systems are closed and proprietary: the protocols are not published; independent client, seed, and tracker implementations are not possible without reverse engineering; there is no forum for discussion and criticism of the various designs; and the companies fully determine what content is distributed over their systems.

As with BitTorrent, when designing a live video system that is both open and P2P, the system must include mechanisms that incentivize peers to contribute upload capacity. But unlike BitTorrent, the incentive cannot be "download the file faster" as there is no notion of faster downloads in live streaming. We advocate a new incentive principle for live P2P streaming, namely, *peers that upload more see higher quality video*. Ideally, peers that free-ride will receive

at best poor quality; peers that upload at a high average rate can receive maximal quality; and peers that upload at more modest rates receive moderate quality. Implicit in this incentive principle is that the system will make available different video qualities. With different video coding schemes, video quality can be defined with different criteria.

In this paper we propose **Substream Trading**, a new P2P streaming design which enables differentiated video quality that is commensurate with a peer's upload contribution. Importantly, the substream trading framework can accommodate different video coding schemes, e.g., single-layer coding, layer coding, and multiple description coding (MDC). Moreover, the design provides a framework for an open P2P live video standard. Substream trading has the following key characteristics:

- *Substream trading:* The design uses a tit-for-tat mechanism based on substream trading. In the baseline design, peers exchange substreams on an even parity basis: if Alice gives Bob exactly n substreams, Bob will reciprocate with exactly n substreams. If peers receive more substreams and correspondingly more useful bits, they can obtain a better video quality. Thus, the more substreams a peer uploads the more substreams it receives and the better the quality. This is the basic mechanism that incentivizes users to upload more to obtain better video quality. Our final design also allows for altruistic behavior, permitting Alice to over reciprocate to Bob when she has spare upload capacity.

- *Mesh design:* Peers self-organize into a mesh as a function of their available bandwidth and content. The mesh overlay is robust and easy to manage in the highly dynamic P2P environment.

- *Substream rather than chunk focus:* Peers notify, select, request and deliver video in basic content units of substreams instead of chunks. As compared to a chunk-based pull-scheme (as in PPLive), the substream design achieves a smaller playback lag with less signaling overhead.

- *Flexibility in video coding:* The design can accommodate different video coding schemes. With single-layer video, the substreams can be generated by time division of the encoded video; while with layer coding or MDC, the substreams are the video layers or descriptions, respectively.

To thoroughly investigate substream trading, we apply it on a single-layer video system and a layered video system. For both systems, the substream trading scheme can provide differentiated video quality and a high overall system performance. In this paper, we make the following contributions:

- We first make the simple, but critical observation that an open P2P live streaming system needs an incentive mechanism, and that the appropriate incentive for streaming is not "download faster" but to get better quality.

- We propose a new P2P streaming design, substream trading, based on a dynamic mesh network (as opposed to a tree design), and trading substreams (rather than trading chunks). To our knowledge, this is the only P2P live streaming framework that supports different video coding schemes and explicitly addresses the incentive issue. The design can be used as a framework for an open P2P live streaming system.

- We examine the possible integration of the proposed substream trading scheme with different video coding schemes: single-layer coding with and without simulcast, layer coding, and MDC.

- Using traces for peer dynamics from a real-world P2P live streaming system, we evaluate the performance of substream trading using both single-layer video and layered video. We show that it is self-scaling, has high efficiency, provides differentiated service, low start-up latency, synergies among peers with different Internet access rates, and protection against free-riders.

The remainder of this paper is structured as follows. Section 2 discusses our fundamental design decisions. Section 3 presents the design of the substream trading system. We consider integration of the substream trading mechanism with different video coding schemes in Section 4, and delineate pros and cons of each. In Section 5, we evaluate the performance of single-layer video and layered video systems based on trace-driven simulations. We discuss related work in Section 6 and conclude in Section 7.

2. FUNDAMENTAL DESIGN CHOICES

For a live P2P video distribution, there is a source (analogous to the initial seed in BitTorrent) and a group of peers watching the video. We refer to the source and the group of peers as a *torrent*. The source encodes and divides the captured video into chunks, and disseminates the video chunks into the P2P torrent. Each peer receives chunks from the source or from other peers (or from both the source and peers). The video source may only have modest upstream Internet access (perhaps less than 1 Mbps), using cable modem, DSL, Wi-Fi, or 3G wireless networks.

2.1 Tree or Mesh?

There is a debate in the literature on which architecture (tree vs. mesh) is more suitable in P2P live streaming. With a tree approach, peers are organized into a single tree or multiple trees [4, 17, 3, 26]. The source pumps video chunks through the trees. The tree structure can be optimized to efficiently disseminate video chunks. However, performance with trees can suffer from peer churn, and trees can be difficult to manage in a highly dynamic P2P environment.

With a mesh approach, peers self-organize into a mesh as a function of their available bandwidth and content. If there is an overlay link between two peers in the mesh, those two peers are said to be partners. In a decentralized fashion, peers form and update partnerships, and explicitly exchange content availability information with their partners. Based on this information, peers select what content to request from their partners. Currently all of the large-scale industrial deployments (PPLive, PPStream, UUSee, and so on) use a mesh design. The most important feature of the mesh approach is that the dynamic overlay is very robust to peer churn, due to the loose relationship among peers. It has further been reported that mesh overlay outperforms tree

or multiple tree from several perspectives [15]. We therefore adopt a mesh overlay in our design.

2.2 Chunk or Substream?

With a mesh overlay, a key design consideration is what is the basic content unit for notifying, selecting, requesting and delivering. One widely adopted approach is to divide a video into chunks, with each chunk consisting 1-4 seconds of video. Such a chunk-based design reduces the dependency of a peer on a particular partner: a peer can request a chunk from any of its partners who has this chunk. This flexibility further increases the robustness of a system to peer churn. Additionally, it allows a peer to use its upload bandwidth with fine granularity. A peer with low upload bandwidth can serve a chunk to its partner, even though it may take a relatively long time. However, this chunk-based design has a playback lag and overhead trade-off [31, 11]. To reduce the playback lag, a peer has to send data availability notifications frequently; otherwise, the lag will normally be large.

Recently, substream-based approaches have been proposed to mitigate this problem [31, 11]. In these proposals, the video is first divided into multiple substreams by simply time dividing a single layer video. For example, assume there are totally S substreams, substream s will contain chunks s, $s+S$, $s+2S$, ..., from the coded video chunk stream. A supplier informs its partners of the substreams it has available. A receiver then determines which substreams should be obtained from which suppliers. When a supplier is assigned to send a receiver a particular substream, it forwards any received chunk belonging to this substream to the receiver immediately, without explicit chunk request notifications. This significantly reduces the playback lag. Additionally, signaling overhead is reduced by batching the notifications of chunks into substreams. As with a chunk-based mesh design, this design is robust to peer churn. The substreams can be very thin so as to efficiently use the upload bandwidth of peers. *In many ways, the substream design provides the best features of trees (which have small playback lags) and chunk-based meshes (which are robust to high churn rates). In this work, we adopt the substream-based mesh approach.*

In [31, 11], the focus is on showing that a substream approach, where single-layer video is time-divided into substreams, can provide superior overhead and lag performance as compared with a chunk-based mesh approach. Incentives and video coding are not considered in [31, 11]. Our contribution is very different. With the ultimate goal of designing a framework for an open P2P live streaming system, we develop a tit-for-tat incentive scheme – substream trading – for a broad-class of substream systems, which include single-layer video, layered video, and MDC. Using real-world traces, we show that substream trading provides incentives through differentiated service.

3. SYSTEM DESIGN

Our goal now is to design an open P2P live streaming system in the context of a mesh-substream design. As discussed in the Introduction, for an open design, it is critical that there be an incentive to encourage peers to upload substreams to other peers. In this section, we describe a design framework that not only provides the desired incentives, but also has the flexibility to support a variety of substream types.

3.1 Mesh Overlay with Substreams

The source encodes a video into S substreams, with the rate of each substream denoted by r. The substreams can be generated by time dividing a single-layer video, by layer coding, by MDC or by some other schemes. Each substream is further divided into chunks of Δ seconds. At any given instant of time, a peer participating in a torrent will receive a subset of the S substreams from one or more other peers in the torrent (including possibly the source); this same peer will redistribute zero or more of the substreams it receives to other peers. In order for a peer to request substreams from other peers, it needs a mechanism for discovering other peers in the torrent (a peer discovery service) and a mechanism for determining which substreams these discovered peers have. Figure 1 shows a simple mesh-substream system with one source, two substreams, and four peers.

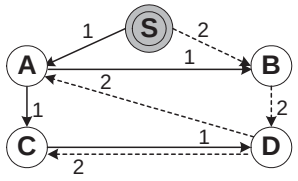


Figure 1: A simple illustration of a mesh-substream system.

As with BitTorrent, peer discovery can be provided by a tracker, in which case the source and all peers participating in the torrent register with the tracker, and discover other peers in the torrent by querying the tracker. As currently done by Azureus BitTorrent clients, peer discovery can also be provided by a DHT, by gossiping peer lists, or by a combination of trackers, DHT and gossiping. The problem of peer discovery is orthogonal to the problems considered here.

After a newly arriving peer P obtains a list of other peers participating in a torrent from the peer discovery service, it contacts peers on the list, searching for *partners*, with whom peer P establishes overlay links. For two partners, it is possible that (i) both serve substream(s) to each other; (ii) only one serves substream(s) to the other; (iii) or neither serves substream(s) to the other. Typically, a peer selects its partners based on some policy, which is referred as a *partner selection problem*.

After having found a sufficient number of partners (on the order of the number of substreams), peer P selects substreams from its partners and the partners sequentially push the video chunks of their selected substreams to peer P. Peer P requests "maps" from its partners periodically, indicating what content they currently have. In BitTorrent, the maps indicate the data chunks that a peer has available for sharing. In our case, the maps indicate the substreams a peer has. For peer P, each of its partners may have more than one substream, and each substream may be available on more than one partner. Given the partners and their substream availabilities, a critical problem of peer P is therefore to determine which substreams should be obtained from which partners. We refer this problem as a *substream selection problem*. If a particular substream is re-assigned to the same partner, peer P will not send the request notification to that partner, and the partner keeps forwarding the chunks from the previously selected substream to peer P.

From time to time, peer P may have to drop partners

that do not have sufficient upload bandwidth or video content, based on some policy. This is referred as a *partnership maintenance problem*. After peer P drops partners, it will try to find replacement partners from the peer list. Note that there are two time scales in this process. In the longer time scale, peers modify their partners; at the shorter time scale, with the set of partners fixed, peers select substreams from their partners.

3.2 Substream Trading

The essence of our scheme is *substream trading*: two partners exchange substreams with each other in a tit-for-tat fashion. In pure tit-for-tat substream trading, peer P sends n substreams to Q if and only if Q sends n different substreams to P. One simple observation is that if all peers use pure tit-for-tat, then each peer receives a number of substreams that is exactly equal to the number of substreams it uploads to other peers; thus, a peer with higher upload contribution is more likely to trade more substreams and eventually obtain better quality. But as we will soon see, pure tit-for-tat is impractical and inflexible; we will need to augment it with mechanisms that address a number of critical issues. In the following sections, we explore these issues in detail.

Peer P and peer Q may trade only one substream (single substream trading) or more than one substream (multiple substream trading) with each other. Multiple substream trading can be considered as a set of single substream trading. For example, if peer P trades two substreams with peer Q, then for peer P, peer Q can be considered to be two *virtual* partners, Q1 and Q2. P trades only one substream with each of the two virtual partners. If the partnership between P and one of the virtual partners breaks up, then peer P and peer Q will simply reduce to the single substream trading. For this reason, in the following sections, we discuss the proposed schemes based on single substream trading.

3.2.1 Partner selection

Recall that the video is encoded into S substreams at the source. We denote the configured maximum upload rate of peer P as bandwidth C, which varies from peer to peer. To simplify notation, assume throughout that C is a multiple of r. Thus, the peer is able to trade up to $T = \min(C/r, S)$ substreams. When a peer is trading less than T substreams and has spare bandwidth, it will search for additional partners for trading. When peer P meets another peer Q that also has spare bandwidth, the two peers should decide whether to establish the partnership. These two peers can simply agree on forming a partnership without a specific policy. However, after forming the partnership, these two peers may not be able to trade substreams properly. For example, in a layered video system, a high bandwidth peer cannot trade with a low bandwidth peer for a high video layer.

To avoid establishing a partnership with an unsuitable partner, peer P may have to pre-screen the candidates before forming the partnership. To this end, peer P can precheck the substream availability of the candidate peer Q, by requesting the substream map from peer Q. Since substream trading is a two-way process and both peer P and peer Q have to make such a decision, they exchange their substream maps for partner selection. With the substream maps from peer Q and its existing partners, peer P can decide whether peer Q can introduce additional substreams or not. This is achieved by using a substream selection algorithm that will be described in Section 3.3.

If both peer P and peer Q have at least one needed substream for each other, they *will* select each other as partners. If not, they have to decide whether to form a partnership. For peer P, if peer Q does not have any substream that peer P needs currently, it does not necessarily mean that Q is not a suitable partner of P. This is because peer Q may obtain the needed substreams of P after a while, from its other partners. There is a dilemma here. If they select each other as partners, they cannot trade immediately; if they don't form a partnership and keep searching, it may be possible for both of them to find partners for immediate trading, but that is costly in terms of time and overhead. Thus, peer P and peer Q should make a decision whether to form a partnership, if they cannot trade substreams immediately.

We make the observation that a peer with fewer substreams, and accordingly less ability for trading, has higher motivation to form such a partnership. This is because a peer with less substreams available may take a long time to find a partner that needs its substreams and form an immediate trading partnership. Following this philosophy, we propose a simple scheme, which we will use in our trace-driven simulation analysis of substream trading. Assume that peer P has s_P substreams and peer Q has s_Q substreams. In this scheme, peer P (peer Q) agrees to form a partnership with probability p_P (p_Q). If both peer P and peer Q agree on forming a partnership, they will establish the partnership; otherwise, they search for new peers. In this scheme, p_X (X represents P or Q) is simply given by:

$$p_X = (\frac{S - s_X}{S})^\gamma, \quad (1)$$

where γ is a control parameter. Note that a newcomer with $s_X = 0$ always agrees on forming such a partnership, since it does not have any substream to trade.

3.2.2 Partnership maintenance

After peer P and peer Q form a partnership, they request substream maps and select substreams from each other. After substream selection, it is possible that peer P and peer Q enter into a *non-trading relationship*, i.e., not both partners have the needed substreams by each other. In our design, if only peer P has a substream that peer Q wants, peer P will deliver it for free. This shares some similarities with "optimistic unchoke" in BitTorrent, where a peer uploads without requiring reciprocation. It addresses the situation where a peer is willing to contribute to the system but does not have any start-up content. If P and Q cannot transfer from the non-trading relationship to a *trading relationship* (i.e., both partners have the needed substreams available by each other) within a time threshold W_n seconds, they will break the partnership. Both peer P and peer Q periodically request substream maps and re-select substreams from each other.

For partners with a trading relationship, both of them monitor the trading procedure with each other. We use a double-sliding window approach to evaluate a partner's performance. With this approach, the peers count the number of downloaded chunks and correspondingly calculate the download rates from their respective partners within each of the two time sliding windows, with one being shorter than the other. Denote the lengths of the sliding windows to be

W_s and W_l, where $W_s < W_l$. We denote the download rates within the two sliding windows as r_s and r_l, respectively. If a partner serves constantly, both rates r_s and r_l should be equal to r. However, due to Internet jitter or a temporary content deficiency, a partner may not be able to deliver with a rate r constantly, especially over a short time scale. To handle this, we introduce tolerance factors for both sliding windows, and consider a partner to provide sufficiently good service quality if $r_s \geq \alpha * r$ and $r_l \geq \beta * r$. The shorter window is used to detect a highly incapable or uncooperative partner quickly. It typically needs a smaller tolerance factor α, e.g., $\alpha = 0.5$. The larger window is used to detect long-term bandwidth or content deficiency. Since time W_l is long enough to smooth out Internet jitter, β should be close to 1.0. If a peer detects that a partner cannot pass the double-sliding window test, it will seek a replacement partner.

3.3 Substream Selection

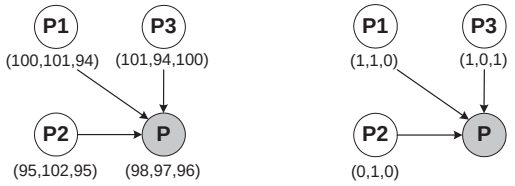


Figure 2: (a)Substream maps; (b)Abstracted substream maps.

In our substream trading system, a peer should determine that which substreams should be obtained from which partners. Before substream selection, the peer periodically requests substream maps from all its partners. Figure 2(a) shows the typical substream maps of peer P and its partners P1, P2, and P3. As an example, the substream map of P1 indicates that it has three substreams, and the sequence numbers of the last chunks from substreams 1, 2, 3 are 100, 101, 94, respectively. Assume that there is no chunk loss during the transmission (e.g., by using the TCP connections for chunk delivery, or by inserting sufficient FEC chunks), the sequence number of the last chunk is sufficient to indicate the chunk availability of a particular substream. For P1, although it has three substreams available, only substreams 1 and 2 can be pulled by peer P, since peer P already has more chunks from substream 3 than P1. Thus, in Figure 2(b), only substreams 1 and 2 are indicated as available in P1. Note that this automatically eliminates possible loops while delivering a substream in a mesh network. We record this processed data availability of partners in *abstracted substream maps*, as shown in Figure 2(b).

With the above definition, we can formulate the substream selection problem as an optimization problem. Assume a peer has N partners $1, \ldots, N$ for requesting substreams. The set of available substreams in partner n is defined as $\mathcal{S}_n$. Let $x_{sn} = 1$ denote that substream s is received from partner n. Since a partner can send at most one substream to a peer (with the virtual partner definition), x_{sn} is subject to the following constraint:

$$\sum_{s \in \mathcal{S}_n} x_{sn} \leq 1, \quad n = 1, \ldots, N.$$

Furthermore, since a substream only needs to be sent from one partner, we have the following constraint as well:

$$\sum_{n} x_{sn} \leq 1, \quad s = 1, \ldots, S.$$

By substream selection, a peer tries to maximize the received video quality. With different video coding schemes, the importance of each substream could be different. For example, with single-layer coding and MDC, the substreams have equal importance, and the peer only needs to maximize the number of received substreams; while with layer coding, the substreams have unequal importance, and the peer needs to take into account the importance of different substreams and maximize the received video quality. To reflect the importance of substreams, we assign weights to each substream, with a larger weight indicating a more important substream. With these weights, the optimal substream selection problem can be converted to maximizing the weighted sum of all substreams as follows:

$$\begin{aligned} \max \quad & \sum_{s=1}^{S} w_s x_{sn} \\ s.t \quad & \sum_{s \in \mathcal{S}_n} x_{sn} \leq 1, \quad n = 1, \ldots, N, \\ & \sum_{n} x_{sn} \leq 1, \quad s = 1, \ldots, S, \end{aligned} \tag{2}$$

where w_s denotes the weight of substream s. This is the classical maximum weight matching problem in a bipartite graph as shown in Figure 3, which can be solved with a complexity of $O(S^3)$ [6]. We will give examples of an appropriate assignment of weights for single-layer video and layered video in Section 4.

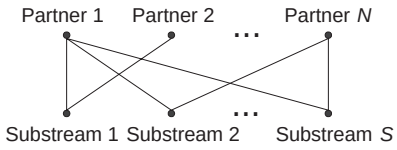


Figure 3: A bipartite graph representing the substream selection algorithm.

3.4 Altruistic Peers

A peer is altruistic at any given time if its aggregate upload rate is higher than its aggregate download rate. With the presence of altruistic peers, bandwidth-deficient peers can possibly receive video at rates higher than their contributions. We do not force a peer to donate bandwidth. We assume that a bandwidth-rich peer will only consider donating if it is receiving all substreams (i.e., the full video rate) and still has surplus upload bandwidth.

Assume a peer is willing to contribute upload bandwidth C where $C/r > S$. This peer can donate $C/r - S$ substreams, i.e., it can provide other peers substreams without reciprocation. In our design, it is the benefactor that determines who will be its beneficiaries. For simplicity, an altruistic peer can randomly select its beneficiaries. A biased donation can also be used. For example, the benefactor can first donate the substreams to its existing partners, and then other peers.

As we will see in Section 5, such a biased scheme helps to combat free-riders.

4. VIDEO CODING

In this section, we show how substream trading can be applied to a variety of different video coding schemes, including single-layer video, layered video, MDC, and simulcast.

4.1 Single-Layer Video

Let us first consider single-layer video, which is currently used by most P2P video applications, including all the popular live P2P industrial deployments, such as PPLive, PPStream, and Coolstreaming. Single-layer video is widely adopted because of its high coding efficiency and its simple design.

With single-layer video, a video is time-divided into S substreams each of rate r. If a peer receives all S substreams, it can reconstruct the video perfectly; otherwise, the peer will reconstruct a corrupted video due to losses of video chunks. One approach to conceal the loss is to simply repeat the latest correctly reconstructed video frame until a new video frame can be reconstructed correctly. (This usually occurs when chunks containing the next I-frame are received, where I-frames are frames that are coded without referencing to previous frames.) This leads to frame freeze and discontinuous video playback. With substream trading, if the upload bandwidth of a peer is higher than the video rate, it can trade all substreams and obtain a continuous video playback; otherwise, it can only trade part of substreams and obtain a less continuous video playback. This provides the basic incentives for peers to contribute upload bandwidth.

Note that not all peers in a single-layer system are necessarily self-supported, with an upload bandwidth higher than the video rate. If no peer contributes at a rate higher than the video rate, the peers with low upload bandwidth cannot receive all S substreams. This may discourage such peers from using the application. But as in BitTorrent, where peers do not always immediately quit after receiving the entire file, we expect to see some altruistic behavior in P2P live streaming [30].

With single-layer video, substreams have equal importance. Thus, the weight for each substream for selection can be defined as $w_s = 1$, $s = 1, \ldots, S$.

4.2 Layered Video

Layer coding encodes a video into several layers with nested dependency, i.e., layer n is only useful if layers 1 through $n-1$ are all available. More received layers reconstruct the video with higher fidelity in terms of amplitude/temporal/spatial resolutions.

Layered video has been suggested for many multimedia client-server applications, mainly to address the bandwidth heterogeneity of the receivers, and the temporal variation of the sustainable receiving rate, due to either Internet congestion or wireless channel fading. However, to date, there has been no widely deployed network application that uses layered video. The primary reason for this is that the original layered codecs were much less efficient than the corresponding single-layer codec – to achieve the same video quality, layered video required a substantially higher bit rate than single-layered video.

In recent years, significant advances have been made in layer coding. Now H.264/SVC (layer coding) achieves a rate-distortion performance comparable with H.264/AVC (single-layer coding), with the same visual reproduction quality typically achieved at +/- 10% bit rate [29]. It is reported that a real-time system with H.264/SVC encoder and decoder has been successfully implemented [28]. Thus, thanks to these recent advances, layer coding is a viable candidate for P2P live streaming systems. Furthermore, layered video is a particularly useful concept for P2P, even more so than for client-server. *In P2P, layered video responds to heterogeneous upload rates as well as heterogeneous download rates and congestion.*

With substream trading, a peer with a higher upload contribution will trade more layers and consequently obtain a better video quality. Furthermore, with layered video, even a small number of layers can lead to passable video quality without discontinuity. Peers with low upload bandwidth can therefore be self-sustaining and less dependent on altruistic peers.

To reflect the layer dependencies, the weights for substream for selection can be set to $w_s = 2^{S-s}$, $s = 1, \ldots, S$. With these weights a lower layer is more important than the sum of all its upper layers, which is consistent with layered coding.

4.3 MDC

Like layered video, MDC generates multiple substreams. But unlike layered video, each of the substreams is of equal importance, so that video quality is only a function of the total number of substreams received and not of which substreams are received. Because all substreams have equal importance, designing a P2P live streaming system using MDC (rather than layered video) is appealing and more straightforward. A number of recent papers have investigated combining a large number of MDC substreams with P2P to create P2P video streaming systems [3, 17, 24, 16]. Like layered video, the proposed substream trading *can* be applied with MDC. In this case, the w_s for each description should be equal.

The efficiency of MDC depends on the trade-off among the achievable qualities with different number of descriptions [27]. MDC is inherently inefficient when a large number of descriptions are created. Among the few proposed methods for generating a large number of descriptions (> 4), MD-FEC [22] together with scalable video coding, and temporal subsampling [8], both introduce significant ($> 60\%$) overhead, compared with single-layer video. *The inefficiency of MDC largely prevents its usage in practical P2P live streaming systems.* For this reason, we do not further consider MDC in this paper.

4.4 Simulcast

With simulcast, the video source encodes a video into multiple independent streams by using single-layer coding, with each stream having a different rate. Each stream then gets distributed within a separate torrent, with no interaction among the torrents. Compared to layered video, simulcast requires more source bandwidth. For example, a set of 5 video simulcast streams, from 200 kbps to 1 Mbps with a 200 kbps step size, would minimally require 3 Mbps at the source; the corresponding layered design would minimally require 1 Mbps. When the sources are residential broadband connections, this becomes a critical issue. When the sources have a sufficient upload capacity to support several

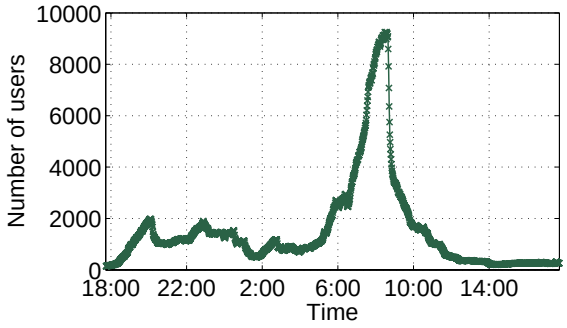


Figure 4: Evolution of number of users viewing a TV channel in the PPLive network.

torrents, the substream trading can be applied within each torrent, as discussed in the single-layer video system. However, such a design would suffer from lack of sharing across the simulcast streams, as we briefly discuss in Section 5.3.

5. TRACE-DRIVEN SIMULATION

We conduct extensive simulations to evaluate the performance of the single-layer system and the layered system with substream trading. We also investigate the impact of cheating behavior.

5.1 Simulation Setup

We developed a packet-level discrete-event simulator in C++. With this simulator, all actions are performed at the packet-level, including the video chunk flow and signaling between peers. In our simulations, we assume that the end-to-end bandwidth bottleneck is at the access links and not in the Internet core. We do not simulate the delay induced by routing in the Internet core; instead, we randomly assign the end-to-end propagation delay between each pair of peers to be between tens of milliseconds to hundreds of milliseconds. Such an abstraction speeds up the simulation and we believe it can still give us an accurate evaluation of the system. Although real Internet experiments in principle could provide more accurate evaluations, it is difficult, if not impossible, to have a controllable environment that can support a large number of concurrent residential peers, which is a key design consideration in P2P networks.

Peer dynamics are simulated by traces collected from a real-world P2P live streaming system – PPLive [20, 9]. 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 was collected from Nov 22nd 17:43, 2006 to Nov 23rd 17:43, 2006, and there were totally more than 100,000 video sessions during this period. Figure 4 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 9000 concurrent users), short video sessions (shorter than one minute), long video sessions (longer than 16 hours), and flash crowds.

In our simulations, the upload bandwidth of the source is set to 2 Mbps. The upload capacities of peers are assigned randomly according to the distribution of Table 1. To come up with an accurate bandwidth distribution of Internet users, we jointly consider the measurement studies in [10] and [7]. The overall distribution of residential peers and Ethernet peers is obtained from [10], while the detailed bandwidth distribution of residential peers is obtained from [7]. We exclude modem peers and ISDN peers due to their low upload and download capacities. Note that the upload capacities of Internet users are highly heterogeneous. Because peers may not be willing to contribute their entire upload bandwidth, in our simulations we assume that the peers only contribute portions of their upload bandwidth for trading, which are indicated in Table 1. For example, the 256 kbps peers contribute 150 kbps for trading.

The playback lag between the peers and the source is set to ten seconds. Every one second, a peer exchanges substream maps with its partners. Accordingly, the peer reselects the substreams based on the most recent substream maps. The substream maps and request notifications can be piggy-backed in the video chunks. The tolerance period W_n for the non-trading relationship is set to ten seconds. We set the monitoring sliding window W_s and W_l to 10 seconds and 30 seconds, and set the tolerance rate factors α and β to 0.5 and 0.9, respectively. In a layered system, it is less likely for two partners in a non-trading relationship to switch to a trading relationship. Parameter γ in (1) is therefore set to one for the single-layer system and is set to three for the layered system. In our simulations, a peer can at most contact eight peers to check if they are the suitable partners during one second. If we let a peer contact more peers simultaneously, it can locate the partners faster. But this will introduce higher overhead.

5.2 Single-Layer System

In this section, we investigate the substream trading system with single-layer video. We begin by assuming all peers in the system follow the proposed protocol, without tampering with the protocol to maximize their own benefits. We then consider free-riding and cheating behavior of peers.

5.2.1 *Differentiated services*

We evaluate the single-layer video system in two scenarios. In the first scenario, the system is underloaded and the supplied bandwidth (i.e., average upload bandwidth of the entire system) is higher than the demanded bandwidth (i.e., the video rate). In the second scenario, the system is overloaded and the supplied bandwidth is lower than the demanded bandwidth. To represent video continuity, we introduce *received chunk ratio*, which is defined as the ratio between the number of received video chunks and the number of encoded video chunks.

With the upload bandwidth distribution shown in Table 1, the average contributed upload bandwidth in the system is about 540 kbps. To investigate both the underloaded and overloaded scenarios, we consider two video rates, 500 kbps and 700 kbps. The encoded videos are time divided into 10 and 14 substreams, respectively, with the rate of each substream being 50 kbps. Each chunk has a size corresponding to $\Delta = 250$ ms. We assume all peers become altruistic if they obtain the video at the full rate. In our simulations, the altruistic peers prefer to donate first to their trading partners.

In Figure 5, we show the CDF of the received chunk ratio for different upload bandwidths. From the ten types of peers (see Table 1), five types of peers are shown in the

Table 1: 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
Contributed upload bandwidth	150	250	300	350	400	500	600	800	1000	1000

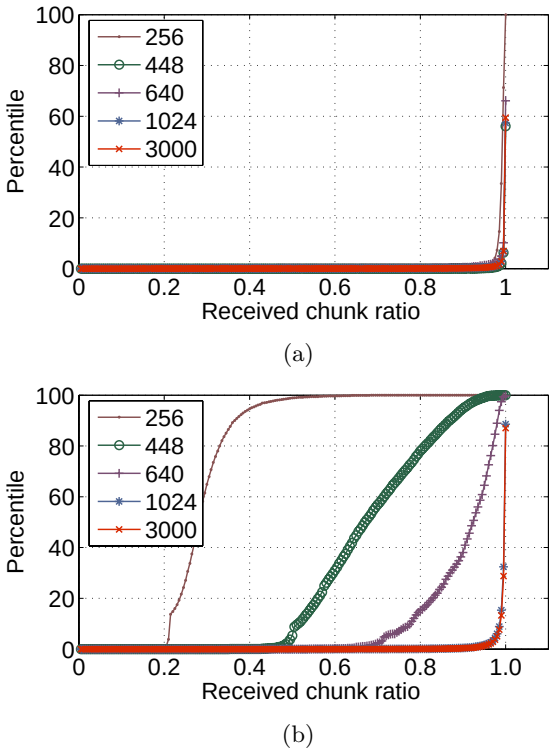


Figure 5: Cumulative distribution of received chunk ratio. (a) Underloaded scenario; (b) Overloaded scenario.

figure, including the peer type with the lowest upload bandwidth (256 kbps), the peer type with the highest upload bandwidth (> 3000 kbps), the peer types with modest upload bandwidth and with many peers (448 kbps and 1024 kbps) and the peer type with the fewest peers (640 kbps). Figure 5(a) shows the results of the underloaded scenario. We observe that almost all peers have a high received chunk ratio that is close to 1.0, indicating that all peers can receive a continuous video quality. But we emphasize that the video qualities of the low bandwidth peers are highly dependent on the altruistic behavior of the high bandwidth peers.

Figure 5(b) shows the CDF of the received chunk ratio under the overloaded scenario. In this case, the upload bandwidth in the system cannot support the video rate for all peers. On average, each peer can at most receive 77% (540/700) of the video chunks. This means that some peers will have very discontinous video quality. With our substream trading design, the peers that have an upload contribution higher than 700 kbps are self-supported and receive continuous video quality. This is verified in the figure, where the peers with 1024 kbps and higher upload bandwidth can receive almost all video chunks. For the peers whose upload contribution is lower than 700 kbps, the received chunk ratio increases with their upload contribution. This incentivizes

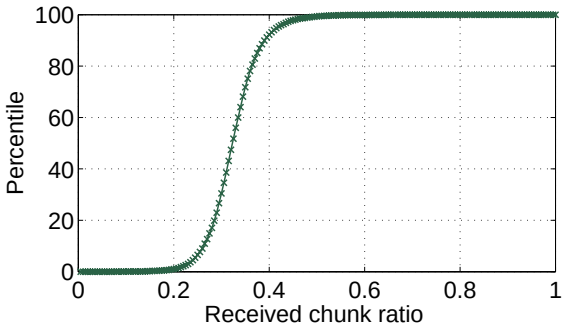


Figure 6: Cumulative distribution of the received chunk ratio for free-riders.

peers to contribute more upload bandwidth.

5.2.2 *Free-riding and cheating*

We now consider free-riding and cheating behavior. As observed in BitTorrent, uncooperative users may tamper with the BitTorrent protocol to maximize their downloading speed [30]. Similarly, in an open P2P live streaming system, a free-rider may try to receive the same video quality as regular peers with minimum upload bandwidth. An open P2P live streaming system should be able to discourage free-riding, by providing minimum video quality for free-riders.

We assume the free-riders try to receive the video rate without contributing any upload bandwidth. In our simulations, we consider one type of cheating, where the free-riders untruthfully announce that they have high upload bandwidth for trading but do not have any content currently. In another words, a free-rider is always pretending to be a newcomer to the system. It is possible for free-riders to establish a partnership with a non-trading relationship with other peers and get served for free during the tolerance time W_n, and then keep jumping around and cheating. We now examine whether a free-rider can receive good video quality under the overloaded scenario. In our simulations, we randomly select 10% of peers as free-riders and assume free-rider establishes partnerships with up to 14 peers.

Figure 6 shows the CDF of the received chunk ratio of the free-riders. We observe that even with cheating, the free-riders get a very low received chunk ratio. Several features of our design assist to discourage free-riders. First, since a free-rider has no content and bandwidth to trade, it will be dropped by its partners after W_n seconds. Second, after a free-rider is dropped, it cannot find replacement partners quickly. As defined in (1), a peer with more substreams available has a lower probability of forming a partnership with a non-trading relationship with another peer. The free-rider will typically need to try several times to find a new partner. With the above two features, it is unlikely for free-rider to simultaneously find a large number (e.g., 14) of partners and receive the full video rate. Furthermore, since altruistic peers prefer to donate spare bandwidth to

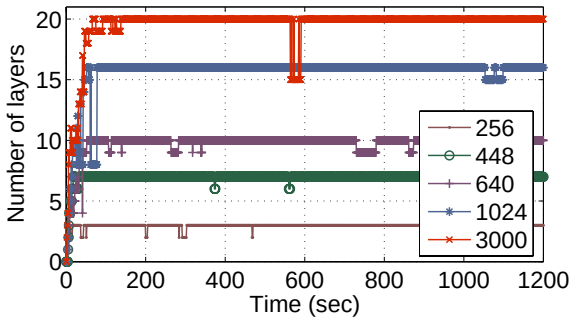


Figure 7: Behavior of typical peers.

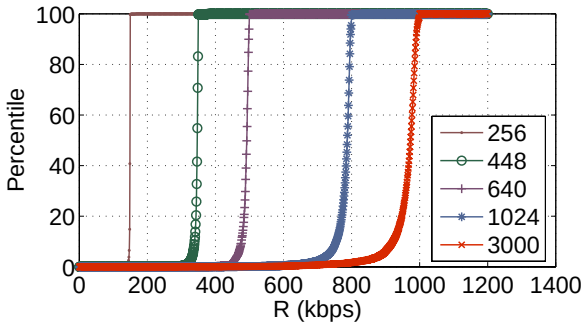


Figure 8: Cumulative distribution of average useful download rate.

their trading partners, it is less likely of a free-rider to get the donation. Additionally, compared with a regular peer, a free-rider has much higher overhead of searching partners and maintaining partnerships.

We nevertheless acknowledge that it may be possible for a free-rider to obtain an acceptable chunk ratio if it continuously establishes partnerships with many peers ($\gg 14$). This will come at the cost of increased overhead. This situation is similar to BitTorrent [12, 14, 19, 23], it appears impossible to fully defend against free-riders in a single-layer system. However, for layered system, we will see even greater robustness to free-riders.

5.2.3 *Summary of single-layer system*

The single-layer video substream trading system has the following properties:

- In an overloaded system, where it is not possible for all peers to get acceptable quality, the peers that upload at rates higher than the video rate do receive all substreams and have maximal quality.
- In an underloaded system where some peers have upload capacity lower than the video rate and others higher than the video rate, the system provides maximal quality to all peers, provided that the high-capacity peers are altruistic.
- Unless free-riders are extremely zealous about cheating, free-riders obtain poor-quality video.

5.3 Layered System

We now investigate substream trading with layered video. In our simulations, the video is encoded into 20 layers, with each layer being 50 kbps and the full video rate being 1 Mbps. For this system, since each peer has a maximum upload rate of 1 Mbps or less, none of the peers can be altruistic.

5.3.1 *Differentiated service*

Figure 7 shows the number of decodable layers of five randomly selected video sessions during their first 20 minutes. We observe that each peer receives a number of layers that is commensurate with its upload contribution. All 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 8 shows the CDF of the received video rates across all video sessions. We observe that almost all peers receive a video rate that is commensurate with their upload contributions. This demonstrates that substream trading provides differentiated services in a layered video system. Furthermore, the system has no bias for the different peer types. For example, the peers with 640 kbps upload bandwidth, which only make up 1.4% of the peer population, also obtain a video quality that is commensurate with their upload contributions.

5.3.2 *Video quality*

The received video rate can largely determine the received video quality. However, in addition to a high received video rate, it is desirable to receive continuous and smooth video quality. Unlike single-layer coding, which needs to receive the full video rate to decode and play a video, layer coding can provide basic video quality if only the base layer is received. For simplicity, we assume the base layer is fully encapsulated in layer 1, and a discontinuity occurs only if the video chunks of layer 1 cannot be received correctly. In our simulation, we observe that all video sessions receive more than 99.99% of video chunks from layer 1; thus, the peers rarely experience playback discontinuity.

With layered video, a receiver may receive varying number of layers, which degrades the user experience. We introduce a smoothness index to evaluate video smoothness in our simulations. The smoothness index is defined as follows:

$$\Phi = \frac{1}{t-1}\sum_{i=2}^{t}\frac{|a(i)-a(i-1)|}{a(i-1)}, \qquad (3)$$

where $a(i)$ is the number of received layers in time slot i, and t is the duration of a video session in terms of time slots. The smoothness index indicates how frequently and dramatically the number of received layers is changing. When Φ is larger, the video is less smooth. In our simulations, we set the length of a time slot equal to the chunk duration Δ.

Figure 9 shows the CDF of the smoothness index of the selected video sessions. We observe that the smoothness index is very low for all peer types. (A smoothness index of 0.005 is extremely low, and most peers have an index much lower than 0.005.) This verifies that peers receive very smooth video quality, which can also be observed in Figure 7. Note that although the highest upload bandwidth peers have slightly larger smoothness index, this does not mean these users see more variation of video quality than the low bandwidth peers. When a large number of layers is received on average, the quality is already very good and

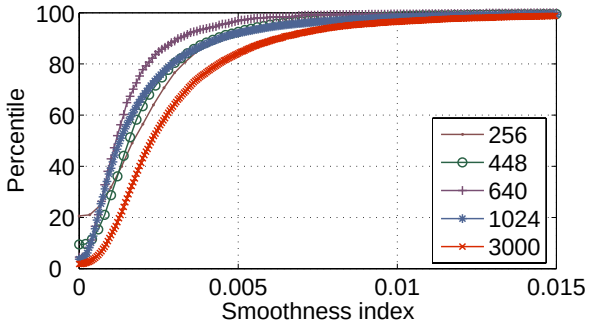


Figure 9: Cumulative distribution of the smoothness index.

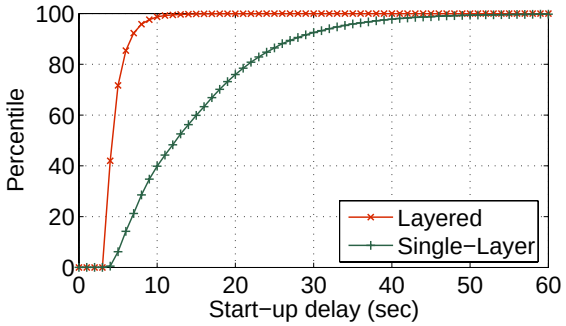


Figure 10: Cumulative distribution of start-up delay across all video sessions.

having slightly more or less number of layers does not produce as much variation in video quality, as in the case where on average a low number of layers are received.

5.3.3 Start-up delay

In P2P live streaming, start-up delay is the time from when a channel is selected until actual playback starts on the screen. This is a critical performance issue, particularly for users who do a lot of channel surfing. Before playback can begin, a peer needs to build an initial reservoir of video chunks to deal with Internet jitter and peer churn. With single-layer video, a peer needs to build the initial reservoir for *all* substreams; but with layered video, the peer only needs to build the initial reservoir for layer 1. In our simulations, if a peer finishes building the reservoir of video chunks for the next three seconds, it starts decoding and playing the video. Figure 10 shows the CDF of the start-up delay for both the single-layer video system (with 500 kbps video rate) and layered video system. We observe that the start-up delay of the layered video system is significantly shorter than that of the single-layer video system. This is because with layered video, only layer 1 is needed to build the initial reservoir and provide passable video quality.

5.3.4 Interaction across peer types

A natural question is whether the resulting layered system essentially creates a stratified system, where peers of the same type primarily share among each other and do not share with peers of other types. To explore this issue, for each peer, we record the download traffic from peers that have higher upload bandwidth (denoted as Higher), peers

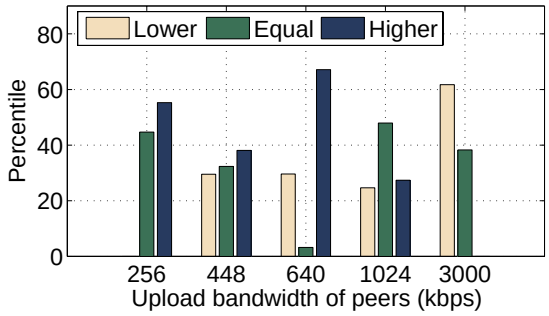


Figure 11: Interaction across peer types.

that have the same upload bandwidth (denoted as Equal), and peers that have lower upload bandwidth (denoted as Lower), and normalize by the peer's total download traffic. We average these values over all peers with the same upload bandwidth, and plot them in Figure 11. We observe that for all five types of peers, there exists a large amount of interaction across peer types. This is especially true for the peer types that have relatively few members, e.g., the peers with 640 kbps upload bandwidth.

Recall that with simulcast, peer types are separated into different simulcast torrents and there is no interaction across peer types. P2P live streaming with layer trading, however, as shown in Figure 11, provides major synergies across peer types. For example, a low-bandwidth peer may serve a high-bandwidth peer with a lower layer. More importantly, with layer trading, a peer type with a small number of members can easily find partners outside its type for trading. This can greatly improve the overall quality of service for the system.

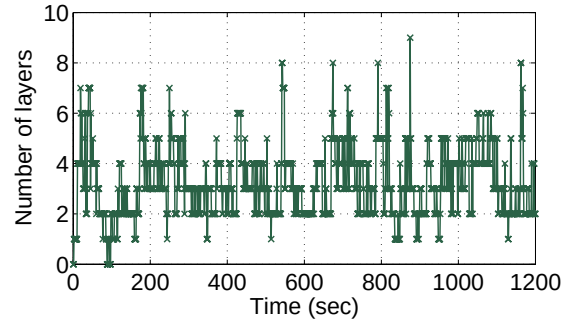
5.3.5 Free-riding and cheating

We investigate free-riding and cheating in the layered video system. We consider the same type of cheating as discussed in the single-layer video system. In order to receive the full video rate (1 Mbps), a free-rider attempts to locate 20 partners simultaneously. Figure 12 shows the video quality of free-riders in the layered video system. Figure 12(a) plots the behavior of a typical free-rider. We observe that free-riders rarely receive video at an average rate higher than 200 kbps. Figure 12(b) shows the CDF of the smoothness index of the free-riders. Most free-riders have a smoothness index that is ten times higher than that of the regular peers, which verifies that the free-riders receive very variable video quality. This is because the free-riders are frequently dropped by their partners. On average, a free-rider is dropped by one of its partners every 0.6 second, leading to a high overhead for locating and managing partners. The low video quality and high overhead cost should largely discourage free-riders.

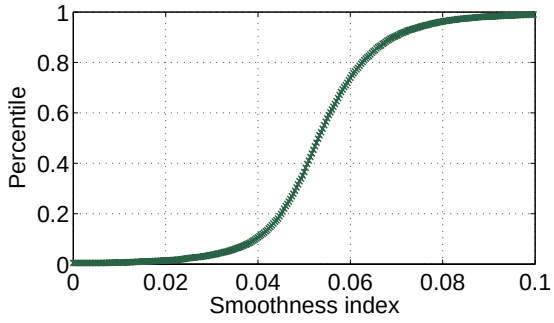
With our proposed partner selection mechanism, it is less likely for a free-rider to establish partnerships with high upload bandwidth peers. Even though a free-rider can locate a large number of partners, most of these partners will be low upload bandwidth peers. With layered video, since these partners only have lower layers, the free-rider can only obtain the lower layers and consequently low video quality. Therefore, the layered system is more robust to free-riders.

5.3.6 Summary of layered video system

The layered video substream trading system has been



(a)



(b)

Figure 12: Video quality of free-riders. (a) Behavior of typical free-riders; (b) Cumulative distribution of smoothness index.

shown to have many desirable properties:

- It provides differentiated service – the video quality that a peer receives is commensurate with its upload rate. Thus peers have an incentive to upload as much as they can.
- For non-freeriding peers, there is little variation in video quality due to fluctuations in the received number of layers.
- The start-up delay is small, and significantly shorter than the start-up delay of single-layer system.
- Peers in different upload bandwidth categories synergistically share layers with each other.
- Aggressive free-riders receive the video at a low rate with relatively high quality fluctuations.

6. RELATED WORK

Over the past few years, there has been a number of proposals for live P2P video in the research community [4, 17, 3, 26, 32]. None of these papers, however, addresses built-in (tit-for-tat) incentives or the design of open P2P streaming systems. Within the context of cooperative peers, Sung *et al.* have recently proposed an MDC-based multiple-tree scheme that uses a novel taxation scheme to provide differentiated services [24]. However, this proposal does not include built-in incentives, assumes cooperative peers, and furthermore uses MDC encoding (which is inherently inefficient as discussed in Section 4.3).

There are three very recent proposals on using tit-for-tat incentives in the context of P2P live video streaming. In a workshop paper, we proposed a tit-for-tat scheme for layered video for *chunk-based systems* [13]. The scheme proposed in this paper has several advantages over that in [13]. First, in this paper we trade substreams rather than chunks, which significantly reduces playback lag and overhead. Second, the framework of this paper can be applied to a variety of coding schemes, including layer coding, MDC, and single-layer coding. Mol *et al.* propose an MDC-based multiple-tree scheme that employs tit-for-tat incentives [16]. 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 inherently inefficient), 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. Finally, Pianese *et al.* propose a chunk-based mesh-pull scheme with single-layer video [18]. The scheme applies a combination of tit-for-tat and donation strategies to provide incentives. In particular, peers with higher upload contribution have more buffered data and are more robust to peer dynamics. However, this scheme is limited to single-layer video and has low throughput.

Unlike [13] [16] [18], the current proposal provides a framework for providing incentives in live P2P video streaming systems. This framework can accommodate a variety of coding schemes. Furthermore, the framework has been optimized for performance, providing differentiated service, high throughput, resiliency to churn, and short start-up delays. The scheme proposed here can serve as a blueprint for an open P2P live video streaming system.

7. CONCLUSION

We have argued that built-in incentives are critical for the design of an open P2P live video streaming system. In this paper, we proposed a framework with live video streaming which has built-in incentives and can accommodate a variety of video coding schemes. In particular, we have shown that substream trading with layered video has many desirable properties, including differentiated service, short start-up delays, synergies across peer types, and protection against free-riders.

We are currently in the process of developing an open-source client that employs substream trading; it can be used with either layered video or with single-layer video with multiple substreams.

8. REFERENCES

[1] http://en.wikipedia.org/wiki/bittorrent_client.
[2] http://torrentfreak.com/5-most-popular-bittorrent-trackers-070924/.
[3] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: High-bandwidth content distribution in a cooperative environment. In *IPTPS*, Berkeley, February 2003.
[4] Yang-Hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *ACM SIGMETRICS*, Santa Clara, June 2000.
[5] Bram Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of*

Peer-to-Peer Systems, Berkeley, June 2003.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithm*. MIT press, 2000.

[7] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu. Characterizing residential broadband networks. In *ACM IMC*, San Diego, October 2007.

[8] Frank H.P. Fitzek, Basak Can, Ramjee Prasad, and Marcos Katz. Overhead and quality measurements for multiple description coding for video services. In *WPMC*, September 2004.

[9] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross. A measurement study of a large-scale P2P IPTV system. In *IEEE Trans. on Multimedia*, volume 9, December 2007.

[10] Cheng Huang, Jin Li, and Keith W. Ross. Can Internet VoD be profitable? In *ACM SIGCOMM*, Kyoto, August 2007.

[11] Bo Li, Susu Xie, Gabriel Y. Keung, Jiangchuan Liu, Ion Stoica, Hui Zhang, and Xinyan Zhang. An empirical study of the Coolstreaming+ system. In *IEEE JSAC*, volume 25, December 2007.

[12] Nikitas Liogkas, Robert Nelson, Eddie Kohler, and Lixia Zhang. Exploiting BitTtorrent for fun (but not profit). In *IPTPS*, Santa Barbara, February 2006.

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

[14] Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. Free riding in BitTorrent is cheap. In *ACM HotNets 2006*, Irvine, November 2006.

[15] Nazanin Magharei, Reza Rejaie, and Yang Guo. Mesh or multiple-tree: A comparative study of live P2P streaming approaches. In *IEEE INFOCOM*, Anchorage, May 2007.

[16] Jan David Mol, Dick H. P. Epema, and Henk J. Sips. The Orchard algorithm: Building multicast trees for P2P video multicasting without free-riding. In *IEEE Trans. on Multimedia*, volume 9, December 2007.

[17] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou, and Kunwadee Sripanidkulchai. Distributing streaming media content using cooperative networking. In *ACM NOSSDAV*, Miami, May 2003.

[18] Fabio Pianese, Diego Perino, Joaquin Keller, and Ernst W. Biersack. PULSE: an adaptive, incentive-based, unstructured P2P live streaming system. In *IEEE Trans. on Multimedia*, volume 9, December 2007.

[19] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in BitTorrent? In *NSDI*, Cambridge, April 2007.

[20] PPLive. http://www.pplive.com/.

[21] PPStream. http://www.ppstream.com/.

[22] Rohit Puri and Kannan Ramchandran. Multiple description source coding using forward error correction codes. In *Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, October 1999.

[23] Michael Sirivianos, Jong Han Park, Rex Chen, and Xiaowei Yang. Free-riding in BitTorrent networks with the large view exploit. In *IPTPS*, Bellevue, February 2007.

[24] Yu-Wei Sung, Michael Bishop, and Sanjay Rao. Enabling contribution awareness in an overlay broadcasting system. In *ACM SIGCOMM*, Pisa, September 2006.

[25] UUsee. http://www.uusee.com/.

[26] Vidhyashankar Venkataraman, Paul Francisy, and John Calandrino. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *IEEE ICNP*, Santa Barbara, November 2006.

[27] Yao Wang, Amy R. Reibman, and Shunan Lin. Multiple description coding for video delivery. In *Proceedings of the IEEE*, volume 93, January 2005.

[28] Mathias Wien, Renaud Cazoulat, Andreas Graffunder, Andreas Hutter, and Peter Amon. Real-time system for adaptive video streaming based on SVC. In *IEEE TCSVT*, volume 17, September 2007.

[29] Mathias Wien, Heiko Schwarz, and Tobias Oelbaum. Performance analysis of SVC. In *IEEE TCSVT*, volume 17, September 2007.

[30] Manaf Zghaibeh and Kostas G. Anagnostakis. On the impact of P2P incentive mechanisms on user behavior. In *NetEcon+IBC*, San Diego, June 2007.

[31] Meng Zhang, Qian Zhang, and Shi-Qiang Yang. Understanding the power of pull-based streaming protocol: Can we do better? In *IEEE JSAC*, volume 25, December 2007.

[32] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. DONet: A data-driven overlay network for efficient live media streaming. In *IEEE INFOCOM*, Miami, March 2005.