SVC-TChain: Incentivizing Good Behavior in Layered P2P Video Streaming

Parisa Rahimzadeh*, Carlee Joe-Wong†, Kyuyong Shin ‡, Youngbin Im *, Jongdeog Lee § and Sangtae Ha *
* University of Colorado Boulder, {parisa.rahimzadeh, youngbin.im, sangtae.ha}@colorado.edu
† Carnegie Mellon University, cjoewong@andrew.cmu.edu
‡ Korea Military Academy, kyuyong.shin@gmail.com
§ University of Illinois at Urbana-Champaign, jlee700@illinois.edu

***Abstract*—Video streaming applications based on Peer-to-Peer (P2P) systems are popular for their scalability, which is hard to achieve with traditional client-server approaches. In particular, layered video streaming has been much-studied due to its ability to differentiate users' streaming qualities in heterogeneous user environments. Previous work, however, has shown that user misbehavior (e.g., free-riding and protocol deviation) poses a serious threat to P2P systems that are not equipped with proper incentive mechanisms. We propose a method to disincentivize such misbehavior. Our SVC-TChain is a layered P2P video streaming method based on scalable video coding (SVC), which uses the recently proposed T-Chain incentive mechanism to discourage free-riding. After introducing T-Chain, we present the first analytical framework to study SVC piece selection with multiple video layers, using it to efficiently choose SVC-TChain's optimal piece selection parameters and thus discourage deviations from the piece selection policy. Extensive experimental results show that SVC-TChain outperforms layered extensions of BiTos and Give-to-Get, two popular P2P video streaming approaches, both in the absence of user misbehavior and when some users misbehave.**

I. Introduction

A recent study [1] showed that IP video traffic made up 70% of all consumer Internet traffic in 2015, a fraction that was predicted to grow to 82% in 2020. Part of this demand is due to commercial peer-to-peer (P2P) live streaming services [2]–[5], which attract hundreds of thousands of daily users. As on-demand and live video streaming become more popular, diverse types of P2P video streaming applications are being developed to support the resulting increase in demand [6], [7].

In the initial stages of P2P video streaming adoption, many content providers preferred to use single layer video coding (in which all participants experience the same video bitrates) such as BiTos [8], Give-to-Get [9], PPStream [3], and PPTV [4]. This approach, however, can only offer users one streaming quality regardless of different users' demands and resource constraints (e.g., different upload and download bandwidth, storage space, or computational power). Layered video coding has been proposed to overcome this disadvantage [10]–[12]. Under layered video coding, individual participants experience different qualities-of-service depending on their available resources. In particular, scalable video coding (SVC [13]), an extension of the H.264/AVC standard [14], is regarded as the *de facto* standard for layered video coding and has received much attention from researchers [15]–[19].

Under SVC, a video is encoded into one base layer and one or more enhancement layers with nested dependency. The base layer provides a base level of quality, while enhancement layers are decoded if and only if the base layer and all lower layers have been received (and decoded). Users' perceived video quality increases with the number of decoded enhancement layers, and they can request and fetch different numbers of layers based on their resource availability. The layers are then split into pieces of equal playback length, which can be used as the units of exchange in a P2P system.

While SVC's flexibility allows it to accommodate different users' needs, it also makes SVC-based P2P streaming heavily dependent on participants' behavior. P2P systems in general are vulnerable to strategic users: if users act so as to maximize their own benefits [20], improperly designed systems may allow them to engage in free-riding, or receiving services with little or no contribution to the system. Many free-riding techniques have proven highly disruptive to BitTorrent-based P2P systems [21], [22], including exploiting altruism [23], cheating [24], the large-view-exploit [25], whitewashing [26], the Sybil attack [27], and collusion [28], and they can also disrupt P2P streaming systems.

To eliminate the destructive effect of free-riding, *we propose SVC-TChain, an SVC-based P2P video streaming scheme that provides high immunity to user misbehavior* and differentiates users' services depending on their resource demands and constraints. SVC-TChain prevents free-riding by adopting a newly proposed incentive mechanism called T-Chain [29]. T-Chain combines Tit-for-Tat direct reciprocity (as used in BitTorrent) with symmetric encryption and indirect reciprocity to nearly eliminate the potential gains from free-riding behaviors.

Eliminating free-riding, however, does not prevent other types of user misbehavior. SVC-based P2P streaming also depends on users' following specified piece selection policies in deciding which pieces to exchange with each other. These policies help to ensure that users receive enough pieces to play back the video on time, which we call *piece sequentiality*– if users cannot obtain the base layer pieces before the given playback time, they will not be able to play back the video. On the other hand, in order for users to exchange pieces, different users must possess different pieces, which may not occur if they all download the video pieces in order of playback.

A poor balance between piece sequentiality and diversity

978-1-5090-5336-0/17/$31.00 ©2017 IEEE

might induce users to deviate from the prescribed policy in order to improve their playback quality, forcing them to find optimal parameter values by trial-and-error and potentially harming compliant users. For instance, if the chosen parameters unduly emphasize piece diversity over sequentiality, users preferring higher video quality may deviate by downloading more higher-layer pieces that are close to being played back. However, their subsequent lack of piece diversity then prevents other, compliant, users from exchanging pieces with them, limiting the range of pieces that compliant users can download. In order to prevent such misbehavior, we introduce the *first analytical work to model sequentiality and diversity in multi-layer SVC piece selection*. Using our framework, SVC systems can efficiently find the optimal piece selection parameters for different system configurations, relieving users of this burden.

We make two main contributions in this paper: in Section II, we propose SVC-TChain, an SVC-based P2P streaming system that nearly eliminates free-riding; and in Section III we introduce the first analytical framework for choosing the SVC piece selection parameters. We then evaluate SVC-TChain experimentally in Section IV, demonstrating its benefits compared with other representative P2P streaming applications. Section V summarizes related work, and Section VI concludes the paper. All proofs are in the [31].

II. SVC-TChain Design and Analysis

In this section, we present the design and analysis of SVC-TChain. We first give an overview of SVC's piece layers in Section II-A before describing the mechanics of users' piece exchanges with T-Chain in Section II-B. We finally outline our piece selection policy in Section II-C.

A. SVC Piece Layers

Scalable Video Coding, or SVC [13], extends the H.264/AVC standard [14] by supporting the encoding of a video file at different spatial (picture resolution), temporal (frame rate), and quality (fidelity or signal-to-noise ratio) scales within one layered bit stream. A video file is encoded at different qualities with one base layer and enhancement layers with nested dependency. Each layer is then partitioned into pieces with equal playback length across the time dimension (Figure 1). Each piece generally takes hundreds of milliseconds to a few seconds to play back [16].

Video sub-streams with different qualities can be derived from a single SVC encoding by dropping some enhancement layers. For example, if a participant $\mathbb{B}$ has downloaded l consecutive layer pieces (from the base to the l^{th} layer) at time t, then $\mathbb{B}$'s streaming rate at time t is $\sum_{i=1}^{l} R(i)$, where $R(i)$ is the streaming rate of the i^{th} layer. Thus, different users can use different numbers of enhancement layers according to their resources (e.g., upload bandwidth, display resolution, computational power, storage space, etc.), guaranteeing efficient and dynamic quality adaptation. For example, a smartphone user with a limited screen resolution and 3G Internet connection may prefer a 15 Hz CIF video stream, while a user with a full HD smart TV and broadband cable connection may prefer a 60Hz HD stream. Users subscribe to a specific target layer out of the total number of available layers, preferentially downloading pieces of layers below this target. Since a user's subscription level represents the hardware resources of their device, we suppose that it's chosen on joining the system and does not change during the streaming.

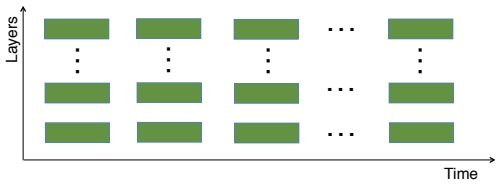
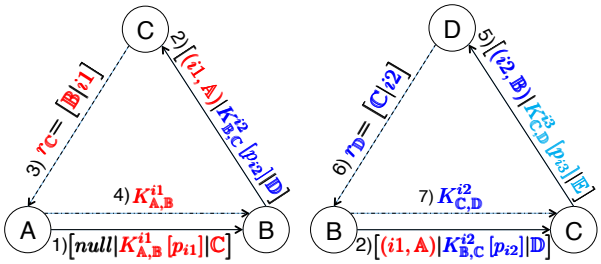


Fig. 1: SVC layers for a streaming application.



(a) First transaction. (b) Second transaction.

(c) Termination.

Fig. 2: Overview of T-Chain.

B. Triangle Chaining (T-Chain)

In SVC-TChain, users exchange pieces of SVC layers via an incentive scheme for cooperative computing called T-Chain [29], which has previously been applied to file sharing applications as an extension of BitTorrent [32]. T-Chain enforces reciprocity between participants, maximizing resource utilization and limiting user misbehavior even under demanding conditions. Figure 2 summarizes T-Chain's operations.

The seeder ($\mathbb{A}$ in Figure 2(a)) begins a chain of transactions by encrypting a file piece with key $K^{i1}_{\mathbb{A},\mathbb{B}}$ and uploading it to a randomly selected requestor $\mathbb{B}$. At the same time, $\mathbb{A}$ informs $\mathbb{B}$ that it must reciprocate by uploading a file piece to a payee user $\mathbb{C}$, who is selected by $\mathbb{A}$ from among $\mathbb{A}$'s neighbors. Requestor $\mathbb{B}$ reciprocates by uploading to payee $\mathbb{C}$ another file piece encrypted with its own key, $K^{i2}_{\mathbb{B},\mathbb{C}}$, which begins the second transaction in the chain. If (and only if) $\mathbb{C}$ notifies $\mathbb{A}$ that $\mathbb{B}$ has reciprocated as requested, $\mathbb{A}$ will release the encryption key $K^{i1}_{\mathbb{A},\mathbb{B}}$ to $\mathbb{B}$, allowing $\mathbb{B}$ to decrypt the piece it received from $\mathbb{A}$ and completing the first transaction.

In the second transaction, $\mathbb{B}$ in Figure 2(b) acts as did $\mathbb{A}$ in the first transaction. Along with uploading an encrypted file piece to $\mathbb{C}$, $\mathbb{B}$ selects a user $\mathbb{D}$ and designates it to $\mathbb{C}$ as the payee to whom $\mathbb{C}$ must reciprocate. If $\mathbb{C}$ is in possession of at least one file piece that $\mathbb{B}$ needs, $\mathbb{B}$ will designate itself as the payee for reciprocation ($\mathbb{D} \equiv \mathbb{B}$), which we call *direct reciprocity*. Otherwise, $\mathbb{B}$ randomly chooses a payee user $\mathbb{D}$ from among its neighbors who need at least one file piece

(including the about-to-be-uploaded piece) held by $\mathbb{C}$. This *indirect reciprocity* thus expands the definition of reciprocation to include (almost) any participant. In the rare case that a sender $\mathbb{X}$ has no neighbor (including itself) who needs to download at least one piece from $\mathbb{Y}$ (Figure 2(c)), $\mathbb{X}$ will upload an *un-encrypted* file piece to $\mathbb{Y}$, releasing $\mathbb{Y}$ from the responsibility to reciprocate and terminating the chain. More details about T-Chain may be found in [29].

C. Piece Selection Policy

Users' piece selection policy defines which pieces recipients request from each other using T-Chain. Since SVC-TChain is designed to support video streaming, the selection policy should (1) ensure seamless video playback and (2) assure a high level of piece diversity for efficient data distribution. Piece diversity is particularly necessary in a flash crowd scenario when many users arrive at once: if all users download pieces sequentially, they will eventually find it difficult to find neighbors with the pieces they wish to download, as no users will have downloaded those pieces in previous time intervals.

While sequential piece selection based on playback deadlines (i.e., downloading pieces earlier if they have sooner playback deadlines) ensures piece sequentiality, it may hurt overall system performance due to a low level of piece diversity. Conversely, a local rarest first (LRF)-like policy can guarantee a high level of piece diversity but may cause intermittent playback pauses. SVC-TChain therefore employs a hybrid of these two policies, with design parameters that can be chosen to tune the balance between piece sequentiality and diversity. We describe the basic policy here before discussing the parameter selection in Section III.

Each user in SVC-TChain maintains three different priority windows (high-, mid-, and low-priority) of pieces based on its current playback position, as seen in Figure 3. The *high-priority window* includes pieces whose playback times are imminent. Thus, the pieces in this window need to be downloaded as quickly as possible, and we use sequential piece selection in this window.[1] The *mid-priority window* has pieces whose playback is not imminent, but will arrive in the near future. The *low-priority window* holds all other pieces. Within the mid-priority and low-priority windows, we use LRF piece selection to improve piece diversity in the system.

We define two parameters, α and β, to determine users' relative emphasis on downloading pieces sequentially or downloading a diverse set of pieces. Users download a piece from the high-priority window with probability α, a piece from the mid-priority window with probability β, and a piece from the low-priority window with probability $1-\alpha-\beta$. While using multiple priority windows is not new [8], [9], *no previous work has provided an analytical method to choose α and β*. While the general effect of α and β is clear–a higher α or β indicates more sequential downloading at the expense of piece diversity–it is not clear which α and β values are optimal. In Section III, we develop the first theoretical framework to evaluate the expected system performance in terms of α and β and find their optimal values.

[1] If the user subscribes to multiple layers, some pieces will have the same playback deadline. We consider this case in Section III-B.

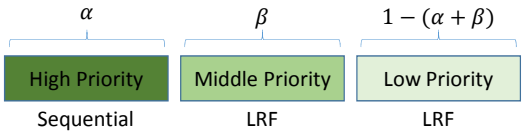


Fig. 3: SVC-TChain piece selection.

III. Discouraging User Misbehavior

SVC-TChain effectively prevents two possible misbehaviors: free-riding and deviating from the piece selection policy by using different α and β values. Incorporating T-Chain (Section II-B) allows us to inherit T-Chain's prevention of free-riding by encrypting downloaded pieces and releasing the encryption key only after reciprocation. This protocol can be shown to dis-incentivize and sharply reduce free-riding, both analytically and in simulation [29] [30].

We focus on choosing the optimal α and β in Section II-C's piece selection algorithm. By developing an analytical framework to choose α and β, we allow an SVC system to choose these parameters without expensive simulations or user trials. We also show that users will have little incentive to deviate from the chosen parameters, preventing a costly trial-and-error period of deviating users searching for their optimal α and β values. We first consider a single-layer video before generalizing the framework to incorporate multiple layers.

A. Piece Selection for a Single-Layer Video

We consider a single-layer video divided into P equal-size pieces, and normalize the units of playback speed so that the video plays for P timeslots. Pieces are indexed by $i = 1, 2, \ldots, P$, and R users arrive in a flash crowd, with one seeder who possesses all of the pieces. We use t_b to denote the buffering time, with high- and mid-priority windows of H and M timeslots respectively. We assume that each user participates in N chains per timeslot and model LRF piece selection within a window as uniform random selection. For each chain, users choose to download from the high-, mid-, or low-priority windows with probabilities $\alpha \geq 0$, $\beta \geq 0$, or $1-\alpha-\beta$ respectively.

Users' objective is to ensure that all pieces are downloaded in time for playback (i.e., sequentiality); thus, we let x_i denote the probability that each piece i will be downloaded by time $i+t_b$. We wish to choose α and β so as to maximize $R\sum_i U(x_i)$, where U is an increasing function. For instance, $U(x_i) = \log(x_i)$ penalizes low probabilities $x_i \approx 0$. However, α and β must also satisfy constraints on piece diversity.

Sequentiality objective. We compute $R\sum_{i=1}^{P} U(x_i)$ by finding the probabilities x_i in terms of α and β.

We first consider pieces in the high-priority window. At time j, the probability that piece $j-t_b+m$ is chosen for download is the probability that m or more chains download high-priority pieces: $\mathbb{P}\left[\sum_{k=1}^{N} \zeta_k \geq m\right]$, where each ζ_k is an i.i.d. random variable that is 1 with probability α and 0 otherwise. Thus,

the total probability that a piece will be downloaded during its H high-priority timeslots is

$$p_H = 1 - \prod_{h=1}^{H} \mathbb{P}\left[\sum_{k=1}^{N} \zeta_k < h\right]. \quad (1)$$

Here we assume a worst-case scenario where no other high-priority pieces have already been downloaded; if others have been downloaded, then p_H is higher than (1).

The probability that a mid-priority piece is downloaded in a given chain is β/M, assuming no mid-priority pieces have already been downloaded. The overall probability that a piece is downloaded while in the mid-priority window is then

$$p_M = 1 - \left(1 - \frac{\beta}{M}\right)^{NM} \quad (2)$$

Finally, the probability that the i^{th} low-priority piece is downloaded at time j is $(1-\alpha-\beta)/(P-H-M-N(j-1))$, where $P - H - M - N(j-1)$ represents the number of un-downloaded low-priority pieces at time j, again assuming no high- and mid-priority pieces have been downloaded. The probability a piece is downloaded while low-priority is then

$$p_L = 1 - \prod_{k=1}^{i+t_b-H-M-1} \left(1 - \frac{1-\alpha-\beta}{P-H-M-N(k-1)}\right)^N. \quad (3)$$

We now use (1–3) to solve for the objective function:

$$R\sum_{i=1}^{H} U\left(1 - \left(1 - \frac{\alpha}{H}\right)^{N(t_b-1)} \prod_{m=H-i+1}^{H} \mathbb{P}\left[\sum_{k=1}^{N} \zeta_k < m\right]\right)$$
$$+ R\sum_{i=H+1}^{H+M} U\left(1 - \overline{p}_H\left(1 - \frac{\beta}{M}\right)^{N(i+t_b-H-1)}\right)$$
$$+ R\sum_{i=H+M+1}^{P/N-t_b} U\left(1 - \overline{p}_H\overline{p}_M\overline{p}_L\right) + R\left(P - \frac{P}{N} + t_b\right)U(1), \quad (4)$$

where $\overline{p}_X = 1 - p_X$. We assume that in the buffering window, pieces 1 to H and $H+1$ to $H+M$ were always designated as high- and mid-priority respectively, with LRF selection.

The maximum value of (4) is $RPU(1)$, which is uniquely achieved at $\alpha = 1$, i.e., purely sequential downloading. In fact, under reasonable conditions, (4) always increases with α:

Proposition 1: For a sufficiently small number of pieces P and $M \geq 2$, (4) is increasing in α for any fixed β and increasing in β for any fixed α. It is minimized when $\alpha = \beta = 0$, corresponding to uniform random selection on all low-priority pieces.

Piece diversity constraints. Piece diversity requirements prevent us from simply taking $\alpha = 1$ to maximize (4). We express these constraints as follows: the probability that user m requests piece i from user n should not exceed the probability that n has previously downloaded piece i. We derive three expressions for this constraint, depending on if piece i is high-, mid-, or low-priority for user m. We consider only $H + M + t_b + 1 < j < (P - H - M)/N$; similar expressions can be derived at the beginning and end of the time interval. If piece i is low-priority at time $j < (P - H - M)/N$, we constrain

$$\frac{1-\alpha-\beta}{P-N(j-1)-H-M} \leq 1 - \prod_{j_1=1}^{j-1}\left(1 - \frac{1-\alpha-\beta}{P-H-M-N(j_1-1)}\right)^N. \quad (5)$$

If piece i is mid-priority at time j, we have the constraints

$$\frac{\beta}{M} \leq 1 - \left(1 - \frac{\beta}{M}\right)^{N(j-(i+t_b-H-M))} \overline{p}_L, \quad (6)$$

using (2) and (3) to represent the probability that piece i has been downloaded while in the low- or mid-priority windows. Finally, if piece i is high-priority at time j, we have

$$\mathbb{P}\left[\sum_{k=1}^{N} \zeta_k \geq i + t_b - j\right] \leq 1 - \prod_{m=i+t_b-j+1}^{H} \mathbb{P}\left[\sum_{k=1}^{N} \zeta_k < m\right] \overline{p}_M\overline{p}_L. \quad (7)$$

Proposition 2: The constraints (5) always hold if $(P-H-M)/N \geq 5$ and $N \geq 2$. If (6) holds for $j = i + t_b - H - M$, then it holds for all $i + t_b - H > j \geq i + t_b - H - M$.

Thus, the piece diversity constraints are most stringent for high- and mid-priority pieces: these are more likely to be requested at time j, so diversity in the piece selection at earlier times is essential to their successful download at time j.

Preventing deviations. We can use the framework above to formally show that users do not benefit by deviating from the specified α and β parameters to obtain more sequentiality. Suppose that a user m downloads all pieces sequentially, i.e., $\alpha = 1$, and that user n downloads pieces according to parameters $\alpha, \beta \in (0, 1)$. Then for $j < H/N$ during the buffering interval, m is likely unable to reciprocate to n:

Proposition 3: Suppose that time $j < H/N < s$ and that user m downloads pieces sequentially. The probability that user n requires a given piece possessed by user m is

$$\alpha \prod_{m=0}^{j-2}\left(1 - \frac{\alpha N}{H - mN}\right). \quad (8)$$

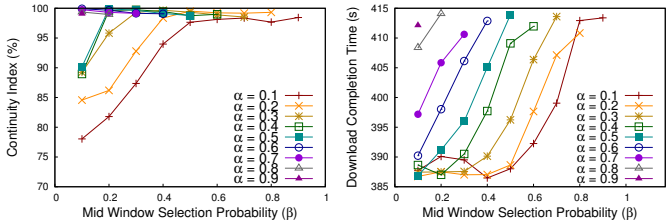
For reasonable values of $H = 10$, $N = 5$, and $\alpha \geq 0.5$, (8) is ≤ 0.18 at $j = 3$: user m is unlikely to be able to exchange this piece with another (compliant) user n.

Choosing α and β. Since we have only two optimization variables, we can use line searches to find the optimal values of α and β, leveraging our objective function's monotonicity (Proposition 1) to reduce the search space. We use Proposition 2 to reduce the number of constraints that we must check to verify the feasibility of given α and β values.

We illustrate the optimal α and β selection by considering 100 users downloading a 2400-piece file. We use a buffering

window of $t_b = 30$, e.g., 30 seconds if each piece represents 1 second of playback, and high- and mid-priority windows of $H = 10$ and $M = 40$, as in previous P2P video streaming studies [9], with $N = 4$. We use $U(x) = \log(x)$ and compute $R\sum_i U(x_i)$ as in (4) for a range of α and β values. The optimal values satisfying (5–7) are $(\alpha, \beta) = (0.55, 0.25)$, for which $R\sum_i U(x_i) = -550$, a 98.7% improvement over the worst case of $\alpha = \beta = 0$ (Proposition 1).

We verify these optimal α and β values by simulating the actions of individual users in a real system with the same H and M values. In this experiment, 100 homogeneous users with 800 Kbps join the system within 10 seconds. One seeder with 2,000 Kbps upload bandwidth stays in the system, while users exit after finishing their video playback. The video file is SVC-encoded with one layer, of 2400 128kb pieces.



(a) Continuity Index (b) Average Streaming Quality

Fig. 4: The continuity indices (a) and the average streaming quality (b) under different α and β values.

Figure 4 shows that the continuity index, defined as the percentage of pieces downloaded in time for playback, increases for each α as the β increase, as expected from Proposition 1. However users' average download completion times increase for each α as the β values increase, reflecting a lack of piece diversity. We therefore wish to find α and β values that maximize the continuity index and minimize the average download completion time. A good choice of (α, β) is (around 0.5, around 0.2), which agrees with our analytical results of $(\alpha, \beta) = (0.55, 0.25)$.

B. Piece Selection with Multiple Layers

If an SVC-encoded video has multiple enhancement layers, we can continue to use LRF in the mid- and low-priority windows to increase piece diversity. However, we must modify our sequential piece selection in the high-priority window to account for the relative importance of the base and enhancement layers. We let l denote the number of enhancement layers that a user tries to download, which we constrain to be $\leq E$, the number of subscribed layers (cf. Section II-A). If all H base layer pieces have been downloaded, the user attempts to download pieces up to layer $l = E$, while if no base layer pieces have been downloaded, the user simply downloads the base layer pieces ($l = 0$) to ensure that the video can be played back. Interpolating between these extremes, we set $l = Eh/H$, where $h \leq H$ denotes the number of high-priority base layer pieces that have already been downloaded.

Sequentiality-diversity analysis. We suppose that α and β are chosen to maximize the time-average of the downloaded video quality (i.e., sequentiality), subject to piece diversity constraints. To formalize this problem, we find the probability that each piece will be downloaded by its playback deadline. Without loss of generality, we assume $t_b = 0$ and consider a user m that participates in N chains.

We index each piece by (i, k), where $i \leq P$ represents the time slot and $k \leq E$ the quality layer, for a total of PE pieces. Letting $p(i, k, j)$ denote the probability that piece (i, k) is downloaded at time j, we have $p(i, k, j) = p_d(i, k, j)p_u(i, k, j)$, where $p_d(i, k, j)$ is the probability that piece (i, k) is downloaded in time j if it is not downloaded already and $p_u(i, k, j)$ is the probability that piece (i, k) is not downloaded until time j. Obviously, $p(i, k, j) = 0, \forall j > i$ and $p_u(i, k, j) = 1 - \sum_{t=1}^{j-1} p(i, k, j)$. We now calculate $p_d(i, k, j)$ when piece (i, k) is in the high-, mid-, and low-priority windows at time j.

If piece (i, k) is in the high priority window at time slot j, it will be downloaded if all previous pieces are already downloaded at time j. Let U^j denote a matrix with random variable elements such that $U^j(i, k) = 1$ if piece (i, k) is not downloaded before time j, and 0 otherwise; u^j denotes a realization of U^j. Given such a realization, piece (i, k) is downloaded in time j if $k \leq l^j$, the number of downloadable enhancement layers at time j, and all previous pieces in the high-priority window have been downloaded:

$$p_d(i, k, j) = \sum_{u^j : k \leq l^j} \mathbb{P}\left(\sum_{n=1}^{N} \xi_n > x\right) \mathbb{P}(U^j = u^j), \qquad (9)$$

where each ξ_n is an i.i.d. random variable that is 1 with probability α and 0 otherwise and $x = \sum_{s=j}^{i-1} \sum_{s'=1}^{l^j} u^j(s, s') + \sum_{s'=1}^{k-1} u^j(i, s')$ is the number of un-downloaded pieces before piece (i, k) at time j. For the sake of simplicity, we have assumed that $U^j(i, k) = 1$ with probability $p_u(i, k, j)$ and 0 otherwise. The number of downloadable enhancement layers at time j is $l^j = E\left(\sum_{t=j}^{min(j+H-1,P)} (1 - U^j(t, 1))\right) / \min(H, P - j + 1)$.

If piece (i, k) is in the mid-priority window, the probability that piece (i, k) will not be downloaded equals $\left(1 - \frac{\beta}{y}\right)^N$, where y is the number of un-downloaded pieces in the mid-priority window. The unconditional probability $p_d(i, k, j)$ that piece (i, k) is downloaded is then

$$\sum_{y=1}^{Emin(M,P-j-H+1)} P_Y(y|y \geq 1) \left[1 - \left(1 - \frac{\beta}{y}\right)^N\right],$$

where Y is the number of un-downloaded pieces in the middle priority window at time j, $Y = \sum_{s=0}^{min(M-1,P-j-H)} \sum_{s'=1}^{E} U^j(j + H + s, s')$, which is the sum of independent Bernoulli random variables; thus, Y has a Poisson Binomial distribution and $P_Y(y)$ is its probability mass function (pmf). If piece (i, k) is in the low priority window at time j, we let $Z = \sum_{s=0}^{P-j-H-M} \sum_{s'=1}^{E} U^j(j + H + M + s, s')$ denote the number of un-downloaded pieces in the low-priority window

at time j and $P_Z(z)$ is its pmf. The unconditional probability that piece (i,k) will be downloaded is then

$$\sum_{z=1}^{E(P-j-H-M+1)} P_Z(z|z \geq 1)\left[1-\left(1-\frac{1-\alpha-\beta}{z}\right)^N\right],$$

Optimization formulation. We take users' objective function to be the average number of downloaded video quality layers at the time of playback, or equivalently the average video resolution; objectives that penalize fluctuations in video quality can also be used. We suppose that if a piece at layer k is downloaded, then all lower layer pieces have also been downloaded, which is ensured by our sequential piece selection for high-priority pieces. Letting $\nu(i,k) = 1 - p_u(i,k,i)$ denote the probability that piece (i,k) is downloaded at the time of playback, we then choose α and β to maximize $\sum_{i=1}^{P}\left(E\nu(i,E) + \sum_{k=1}^{E-1} k\left(\nu(i,k) - \nu(i,k+1)\right)\right)/P$.

We express our piece diversity constraint as follows: the fraction of pieces for which the probability that user m requests piece (i,k) from user n exceeds the probability that n has previously downloaded piece (i,k) should be less than a small value ϵ, i.e.,

$$\frac{\sum_{(i,j,k)} \mathbb{P}\left(p(i,k,j) \geq \sum_{t=1}^{j-1} p(i,k,t)\right)}{EP(P+1)/2} \leq \epsilon, \qquad (10)$$

where $EP(P+1)/2$ is the total number of piece pairs over which we check the condition.

Choosing α and β. As in the single-layer case, since α and β are our only optimization variables, we can use line searches to find their optimal values. We compare the analytically optimal α and β values to those obtained with experimental simulations. Figure 5 shows the average number of downloaded video layers (i.e., the value of our objective function) for different values of α and β. We have assumed $P = 600$, $H = 4$, $M = 16$, and $E = 5$. As can be seen from Figure 5, our analytical results yield many candidate α and β values that result in a high average number of video layers (e.g., $(\alpha,\beta) \in$ {(0.5,0.3), (0.5,0.4), (0.5,0.5), (0.1,0.7),(0.1,0.8), (0.1,0.9)}); all of these satisfy the diversity condition (10) for $\epsilon = 0.01$. Though all of these choices yield a continuity index over 98, we select the one with the highest continuity index, which is $(\alpha,\beta) = (0.5, 0.3)$. The experimental results confirm this selection.

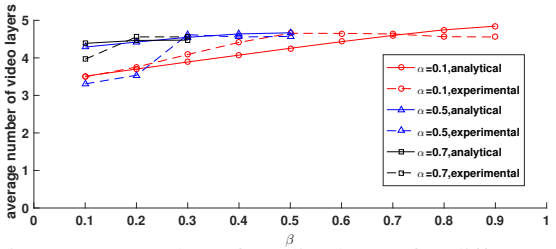
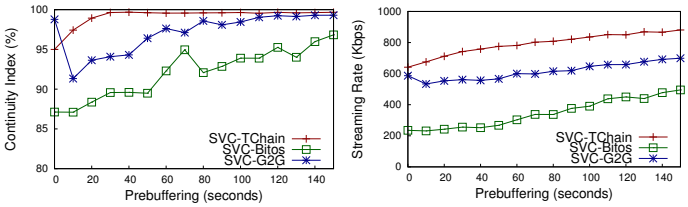


Fig. 5: Average number of quality layers for different values of α and β.



(a) Continuity Index. (b) Average Streaming Rate.

Fig. 6: Performance of compliant users under flash crowd arrivals with no misbehavior.

IV. Evaluation

We evaluate SVC-TChain in a wide range of scenarios by modifying a BitTorrent simulator to incorporate SVC streaming into BiTos [8], Give-to-Get (G2G) [9], and T-Chain [29].

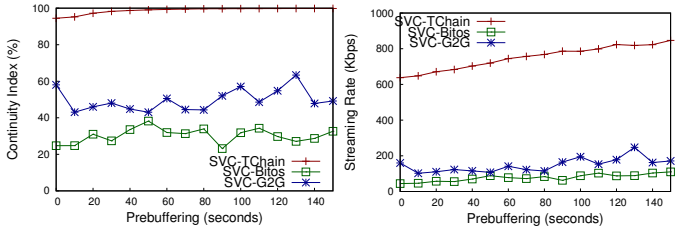
Each experimental run begins with one seeder in the swarm. Users arrive and start downloading the SVC-encoded video file from the seeder and other users, exiting the swarm immediately after finishing their video playback. The size of the SVC-encoded video file is 96 MB, encoded in 1 base and 9 enhancement layers [12]. Each layer is encoded at a rate of 100 Kbps with 600 pieces of 128Kb. The aggregate streaming rate is 1 Mbps, and the total video playback time is 768 seconds. We use $\alpha = 0.5, \beta = 0.3$ for the piece selection, as found using Section III-B's optimization framework.

We assume that each user joining the system waits for a fixed amount of pre-buffering time t_b before starting its video playback. We use two peer arrival models: a *flash crowd* model in which 400 users join the swarm within the first 10 seconds, similar to a live stream, and a *real trace* model based on a five-month RedHat 9 Torrent tracker trace [33]. In the flash crowd model, we evaluate the average performance of all 400 peers, while in the real trace model, we exclude the first 500 of 1000 compliant users to focus on the steady state performance. We compare SVC-TChain's performance to that of SVC-BiTos and SVC-G2G in terms of the continuity index, average uplink utilization, average streaming rate, and normalized fairness factor, which we define as the ratio of the bandwidth contributed to that received.

A. Discouraging Free-Riders

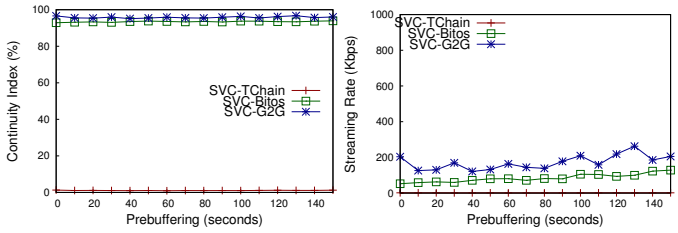
Figure 6 shows the continuity index and average streaming rate of compliant users under the flash crowd model without misbehavior. Both quantities are proportional to the pre-buffering time t_b: with larger t_b, users have more time to download pieces before the playback deadlines. Downloading more base and enhancement layer pieces improves the continuity index and average streaming rate respectively. SVC-TChain significantly outperforms both SVC-BiTos and SVC-G2G in terms of the continuity index and average streaming rate: users in SVC-TChain utilize both direct and indirect reciprocity at the same time, while users in the other two methods use only direct (SCV-BiTos) or only indirect reciprocity (SVC-G2G).

We now repeat the same experiment, but with 20% of the users attempting to free-ride. The amount of available



(a) Continuity Index. (b) Average Streaming Rate.

Fig. 7: Performance of compliant users under flash crowd arrivals with 20% free-riding.



(a) Continuity Index. (b) Average Streaming Rate.

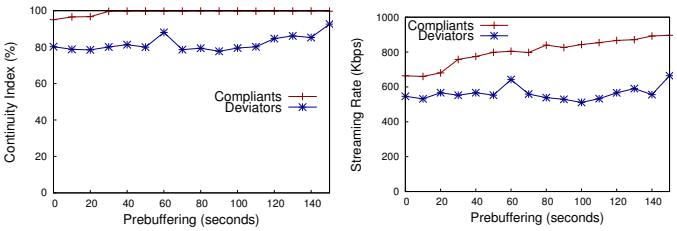
Fig. 8: Performance of free-riders under flash crowd arrivals with 20% free-riding.

resources for each method therefore decreases, since free-riders do not offer their resources to the system. Figures 7 and 8 show the continuity indices and average streaming rate of compliant users and free-riders respectively. Compared to Figure 6, the compliant users in SVC-BiTos and SVC-G2G experience a sharp decrease in the continuity index and average streaming rate, while free-riders can enjoy high continuity indices and rates of hundreds of kbps. Since these methods are not equipped with mechanisms to prevent strategic free-riding, system resources are easily allotted to free-riders, decreasing compliant users' access to system resources. Compliant users' performance in SVC-TChain, however, is virtually unaffected, due to T-Chain's discouragement of free-riding. In fact, it is nearly impossible to free-ride under SVC-TChain: as seen in Figure 8, both the continuity index and the average streaming rate of free-riders in SVC-TChain are almost zero.

B. Discouraging Deviators

We next consider the effect of SVC-TChain users who misbehave not by free-riding, but by deviating from the piece selection policy. For instance, greedy users may prefer to download only pieces in their high priority windows (i.e., $\alpha = 1$) in order to improve their continuity index and streaming rate compared to the policy described in Section II-C.

We compare deviators' performance to that of compliant users under the experimental conditions of Figure 6, with 10% of users deviating by using sequential piece selection ($\alpha = 1$) instead of $(\alpha, \beta) = (0.5, 0.3)$. Figure 9 shows the continuity index and streaming rate for deviators and compliant users. Deviators visibly suffer compared to compliant users: they have less piece diversity, preventing them from the reciprocation that is required for users to download pieces in SVC-TChain. Compliant users, however, maintain continuity indices and streaming rates comparable to those in Figure 6.



(a) Continuity Index. (b) Streaming Rate.

Fig. 9: Performance of deviators when 10% of the total users use purely sequential piece selection.

Their α and β values balance piece sequentiality and diversity, protecting their performance from deviators' misbehavior.

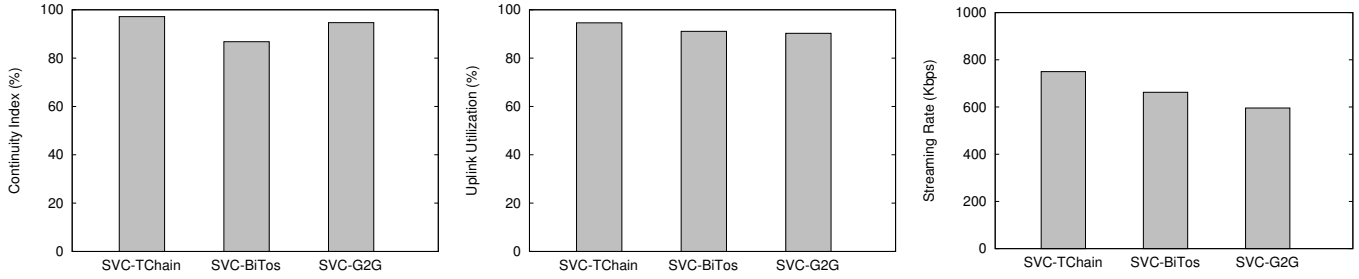
C. Real Swarm Performance

We next investigate the performance of each method under the real trace arrival model. Figure 10(a) shows users' continuity indices without user misbehavior. All three methods perform well, providing users with nearly seamless video playback. SVC-BiTos performs worse due to its piece selection policy: after video playback starts, SVC-BiTos applies LRF in the high priority window while SVC-TChain and SVC-G2G prefer pieces with imminent playback deadlines, sacrificing some piece diversity for sequentiality. Yet Figure 10(b) shows that all three methods have average uplink utilizations over 90%: SVC-G2G's and SVC-TChain's piece diversity is still sufficient for users to actively exchange pieces. With flash crowd arrivals (Figure 6), SVC-TChain's direct and indirect reciprocity yields a higher uplink utilization and average streaming rate (Figure 10(c)) than SVC-BiTos or SVC-G2G.

Figure 11 shows our performance results with real trace arrivals when 20% of the users are free-riders. Surprisingly, compliant users with all three methods experience similar performance as in Figure 10 without misbehavior. In the real trace model, the system resources are plentiful, so 20% free-riding does not significantly affect performance. Compliant users' average uplink bandwidth in the real trace model is around 1,250 Kbps with an average uplink utilization around 90% for all methods (Figure 11(b)), yielding an average upload rate above the maximum streaming rate of 1 Mbps and providing a resource surplus that free-riders can exploit.

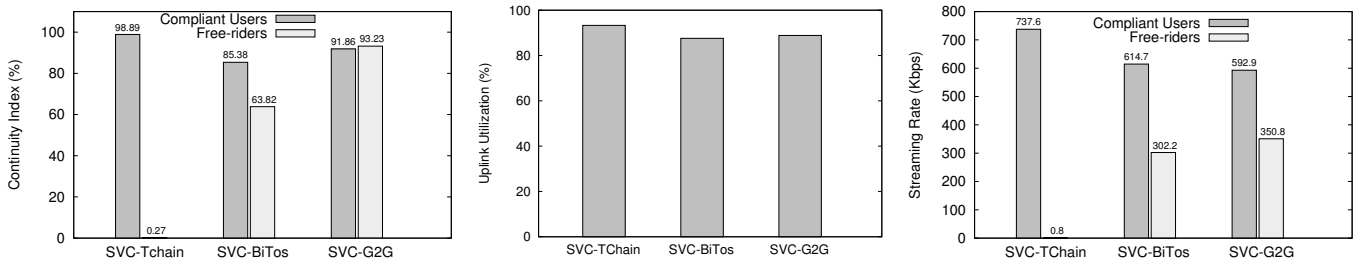
However, free-riding can still harm streaming applications. Free-riders in SVC-BiTos and SVC-G2G can stream at a rate of more than 300 Kbps without any resource contribution (Figure 11(c)). More importantly, free-riders' continuity index in SVC-G2G is higher than that of compliant users, even though compliant users contribute resources and free-riders do not. Free-riders' good performance incentivize others to free-ride, eventually resulting in system collapse. Unlike SVC-BiTos and SVC-G2G, free-riding attempts in SVC-TChain are effectively prevented, as seen in Figures 11(a) and 11(c).

SVC-TChain not only discourages free-riding, but can actively encourage compliant users to join the system. Users in a cooperative distributed system are more likely to contribute resources to a *fair* system in which they receive benefits in



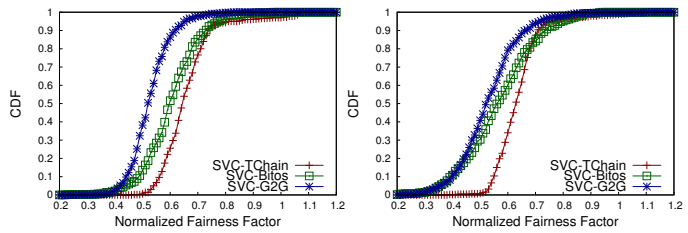
(a) Continuity Index. (b) Average Uplink Utilization. (c) Average Streaming Rate.

Fig. 10: Performance of compliant users under the real trace arrival model with no misbehavior.



(a) Continuity Index. (b) Average Uplink Utilization. (c) Average Streaming Rate.

Fig. 11: Performance of compliant users and free-riders under the real trace arrival model with 20% free-riding.



(a) No free-riders. (b) 20% free-riders.

Fig. 12: Fairness enforced by each method.

proportion to the contribution they make. We show that SVC-TChain is more fair than SVC-BiToS or SVC-G2G both without misbehavior and with 20% of users acting as free-riders. Figures 12(a) and 12(b) show the Cumulative Distribution Function (CDF) of the normalized fairness factor over 500 compliant users without and with free-riding respectively. In a perfectly fair system, a user would receive exactly as much bandwidth as it contributes, and the normalized fairness factor would be 1.[2] We see that SVC-TChain is fairer than SVC-BiToS or SVC-G2G with or without free-riding, incentivizing compliant users to contribute their resources.

V. Related Work

The first video streaming systems based on cooperative computing relied on single layer video coding, giving all users the same streaming quality. Their P2P streaming architectures allowed them to take advantage of P2P's robustness to churn, high scalability, and simplicity of management [12], [34]. Indeed, BitTorrent [32], [35], one of the most popular P2P systems for content distribution, has been widely adapted to video streaming in academic (e.g. BiTos [8], Give-to-Get [9]) and commercial (e.g., PPStream [3] and PPTV [4]) contexts. The piece and neighbor selection mechanisms of BitTorrent should be modified to support streaming, and much work has focused on how to find the optimal modifications [36]–[38]. Little work, however, has investigated how to deal with user misbehavior (e.g., free-riders or deviators) in spite of the fact that these represent significant weaknesses in BitTorrent [24], [29]. The authors of [39] show that layered P2P video streaming provides differentiated service and protection against free-riders in BitTorrent-based P2P live streaming, but do not consider strategic manipulation attacks [24], [29]. To the best of our knowledge, SVC-TChain is the first method for SVC-based layered video streaming that prevents both free-riding and deviations from specified piece selection policies.

Many systems have proposed versions of layered video coding in P2P systems [34]. The advent of H.264/SVC [13], in particular, has drawn much interest from researchers [11], [12], [16], [18], [39], [40]. LayerP2P [11] proposed a 3-stage data scheduling scheme, which is similar to our priority windows. Zigzag scheduling [40] proposed to use a sliding window of pieces with a zigzag style selection. LayeredP2P and Zigzag scheduling, however, do not consider how to ensure piece diversity or prevent user misbehavior.

The authors of [16] shed some light on the impact and tradeoffs of quality adaptation in SVC-based layered video streaming. Their quality adaptation is complementary to our work and can be easily plugged into SVC-TChain to improve

[2]The normalized fairness factor of compliant users for each method is less than 1, which means that the users' streaming rates are lower than their upload rates in the systems. This is mainly due to pieces downloaded after their playback deadlines or un-decodable enhancement layer pieces.

its performance. Some possible piece segmentation and reference challenges for SVC specifically, which are beyond the scope of this paper, are discussed in [18], [41] in detail.

VI. Conclusions

In this paper, we present SVC-TChain, an SVC-based P2P video streaming scheme that offers immunity to misbehavior (e.g., free-riders and deviators) and differentiates users' services depending on their resource demands and constraints. SVC-TChain prevents misbehaviors by adopting a newly proposed incentive mechanism called T-Chain and analytically choosing piece selection parameters that trade off between sequentiality and piece diversity. Through extensive simulations, we demonstrate the SVC-TChain outperforms other representative P2P schemes (SVC-BiTos and SVC-G2G) with and without user misbehavior (i.e., free-riders and deviators). SVC-TChain represents a practical P2P-based streaming protocol that can help address today's ever-expanding demand for video streaming.

References

[1] Cisco, "The zettabyte era–Trends and analysis," 2016, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

[2] X. Zhang, J. Liut, B. Lis, and T.-S. P. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *IEEE INFOCOM*, 2005, pp. 2102–2111.

[3] PPStream, "http://www.ppstream.com/," 2016.

[4] PPTV, "http://www.pptv.com/," 2016.

[5] J. Goldman, "BitTorrent looks to disrupt TV news starting at RNC," CNet, 2016, http://www.cnet.com/news/bittorrent-looks-to-disrupt-traditional-tv-news-starting-at-republican-national-convention/.

[6] X. Su and S. Dhaliwal, "Incentive mechanisms in p2p media streaming systems," in *IEEE Internet Computing*, vol. 14, no. 5, 2010.

[7] X. Kang and Y. Wu, "Incentive mechanism design for heterogeneous peer-to-peer networks: A stackelberg game approach," *IEEE Trans. on Mobile Computing*, August 2015.

[8] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "Bitos: Enhancing bittorrent for supporting streaming applications," in *IEEE INFOCOM*, 2006.

[9] J. J. D. Mol, J. A. Pouwelse, M. Meulpolder, D. H. J. Epema, and H. J. Sips, "Give-to-get: free-riding resilient video-on-demand in p2p systems," in *Multimedia Computing and Networking*, 2008.

[10] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Using layered video to provide incentives in p2p live streaming," in *Proc. of P2P-TV Workshop*, 2007, pp. 311–316.

[11] X. Xiao, Y. Shi, Y. Gao, and Q. Zhang, "Layerp2p: A new data scheduling approach for layered streaming in heterogeneous networks," in *IEEE INFOCOM*, 2009.

[12] H. Hu, Y. Guo, and Y. Liu, "Peer-to-peer streaming of layered video: Efficiency, fairness and incentive," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 21, no. 8, pp. 1013 – 1026, 2011.

[13] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103 – 1120, 2007.

[14] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[15] P. Baccichet, T. Schierl, T. Wiegand, and bernd Girod, "Low-delay peer-to-peer streaming using scalable video coding," in *Packet Video 2007*, 2007, pp. 173 – 181.

[16] O. Abboud, T. Zinner, K. Pussep, S. Al-Sabea, and R. Steinmetz, "On the impact of quality adaptation in svc-based p2p video-on-demand systems," in *Proc. of ACM MMSys*, 2011, pp. 223–232.

[17] R. Meier and R. Wattenhofer, "Peer-to-peer streaming in heterogeneous environments," *Signal Processing: Image Communication*, vol. 27, no. 5, pp. 457–469, 2012.

[18] J. Song, X. Zhou, Y. Zhang, H. Tang, F. Bai, and S. Ci, "A playback length changeable chunk scheduling algorithm for svc based p2p streaming systems," in *Proc. of IEEE GLOBECOM*, 2012.

[19] M. Eberhard, A. Palo, A. Kumar, R. Petrocco, L. Mapelli, and M. Uitto, "Nextsharepc: an open-source bittorrent-based p2p client supporting svc," in *the 3rd Multimedia Systems Conference*, 2012, pp. 101–106.

[20] F. Qin, L. Ge, Q. Liu, and J. Liu, "Free riding analysis of peer-to-peer streaming systems," *Computational Information Systems*, vol. 7(3), pp. 721–728, 2011.

[21] P. Shipley and V. Kumar, "Survey on incentive mechanism," *Global Journal of Computer Science and Technology*, vol. 13, no. 5, 2013.

[22] W. Wu, R. T. Ma, and J. C. Lui, "Distributed caching via rewarding: An incentive scheme design in p2p-vod systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 612–621, March 2014.

[23] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in bittorrent?" in *USENIX NSDI*, May 2007.

[24] K. Shin, D. S. Reeves, and I. Rhee, "Treat-before-trick : Free-riding prevention for bittorrent-like peer-to-peer networks," in *IEEE IPDPS*, Rome, Italy, May 2009.

[25] M. Sirivianos, J. H. Park, R. Chen, and X. Yang, "Free-riding in bittorrent networks with the large view exploit," in *International Workshop on Peer-to-Peer Systems*, 2007.

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

[27] R. Landa, D. Griffin, R. G. Clegg, E. Mykoniati, and M. Rio, "A sybilproof indirect reciprocity mechanism for peer-to-peer networks," in *Proc. of IEEE INFOCOM*, 2009.

[28] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," in *ACM Sigecom Exchanges*, vol. 5, July 2005.

[29] K. Shin, C. Joe-Wong, S. Ha, Y. Yi, I. Rhee, and D. Reeves, "T-chain: A general incentive scheme for cooperative computing," in *IEEE ICDCS*, June 2015.

[30] C. Joe-Wong, Y. Im, K. Shin and S. Ha, "A Performance Analysis of Incentive Mechanisms for Cooperative Computing," in *IEEE ICDCS*, June 2016.

[31] P. Rahimzadeh, C. Joe-Wong, K. Shin, Y. Im, J. Lee, and S. Ha, "SVC-TChain: Incentivizing good behavior in layered P2P video streaming," 2016, http://www.andrew.cmu.edu/user/cjoewong/SVC-TCHAIN-INFOCOM.pdf.

[32] B. Cohen, "Incentives build robustness in bittorrent," in *the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[33] M. Izal, G. Uroy-Keller, E. Biersack, P. A. Felber, A. A. Hamra, and L. Garces-Erice, "Dissecting bittorrent: Five months in a torrent's lifetime," *Lecture Notes in CS*, vol. 3015, pp. 1–11, 2004.

[34] B. Li, Z. Wang, J. Liu, and W. Zhu, "Two decades of Internet video streaming: A retrospective view," *ACM Trans. on Multimedia Computing, Communications, and Applications*, vol. 9, no. 33, October 2013.

[35] B. Cohen, "The bittorrent protocol specification," Feburary 2012. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html

[36] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson, "Analysis of bittorrent-like protocols for on-demand stored media streaming," in *Proc. of ACM SIGMETRICS*, 2008, pp. 301–312.

[37] B. Fan, D. G. Andersen, M. Kaminsky, and K. Papagiannaki, "Balancing throughput, robustness, and in-order delivery in p2p vod," in *Proc. of ACM CoNEXT*, 2010.

[38] Z. Ma, K. Xu, J. Liu, and H. Wang, "Measurement, modeling and enhancement of bittorrent-based vod system," *Computer Networks*, vol. 56, p. 11031117, 2012.

[39] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "Layerp2p: Using layered video chunks in p2p live streaming," *IEEE Trans. on Multimedia*, vol. 11, no. 7, pp. 1340–1352, December 2009.

[40] Y. Ding, J. Liu, D. Wang, and H. Jiang, "Peer-to-peer video-on-demand with scalable video coding," *Computer Communications*, vol. 33, no. 14, p. 15891597, 2010.

[41] O. Mokryn, A. Platner, I. David, and O. Amir, "H.264 svc extension for peer to peer schemes," in *Proc. of IEEEI*, 2012.