Mechanism Design for Dynamic P2P Streaming

Guibin Tian*,Yang Xu*, Yong Liu*, and Keith Ross†
*Electrical & Computer Engineering, Polytechnic Institute of NYU, Brooklyn, NY 11201
† Computer Science and Engineering, Polytechnic Institute of NYU, Brooklyn, NY 11201

***Abstract*—In dynamic streaming, a user can dynamically choose from different versions of the same video. In *P2P dynamic streaming*, there is one P2P swarm for each version, and within a swarm, peers can share video chunks with each other, thereby reducing the server's bandwidth cost. Due to economy of scale, cooperation among peers can also reduce the per-peer content price. In this paper, we use cooperative game theory to dynamically assign each peer to a version. To maximally incentivize peer cooperation, we use mechanism design to develop pricing schemes that reflect content and bandwidth cost savings derived from peer cooperation. With this approach, each peer is assigned to a swarm that is commensurate with its upload contribution and the price it is willing to pay. We also develop and simulate a distributed dynamic P2P streaming algorithm, consisting of chunk scheduling, token-based accounting, and video version switching, to dynamically adjust each peer's video version based on the collaborative behaviors of all peers.**

I. Introduction

The newest trend of video streaming on the Internet is *dynamic adaptive streaming*, where a video is encoded into multiple versions, with each version delivered at a different bit rate. When starting a video session, a user selects a version to stream to match the available resources on his device, such as network bandwidth, screen rendering capability, and current battery level, etc. After the session starts, the user can further switch versions as his resource availability changes [1]. Dynamic adaptive streaming has been widely deployed to provide uninterrupted video streaming service to users [2], [3], [4]. While all deployed solutions to date are server-based [5], [6], [7], the Peer-to-Peer (P2P) paradigm has great potential to enhance the dynamic streaming service. In *P2P dynamic streaming*, there is one P2P swarm for each version, within a swarm, peers watching the same version share video chunks with each other. P2P video sharing reduces the server's bandwidth cost and makes high-quality video versions more accessible when the server has limited capacity. In addition, due to *economy of scale*, cooperation between peers also provides them an edge in negotiating lower content prices with content providers.

The key design question for P2P dynamic streaming is: *which version should each peer watch?* In server-only dynamic streaming, each user's own resources largely determine which version the user watches. In P2P dynamic streaming, video version selections on all peers are tightly coupled. Isolated version selections by individual peers are not sustainable, and cannot fully exploit the cooperation gain. As with any P2P system, the success of P2P dynamic streaming hinges on peers pooling their resources and cooperating. Intuitively, to provide incentives for cooperation, peers contributing more resources should be rewarded by receiving a high-quality version.

In this paper, we formally investigate the incentive issues of P2P dynamic streaming under a *cooperative game theory* framework. Cooperative game theory systematically studies mechanisms that encourage and sustain cooperation among self-optimizing agents. In a traditional cooperative game, each agent is a potential customer for a service, for which there is a charge for receiving the service. Collaboration among agents can potentially reduce individual service charges. With a properly designed pricing mechanism, maximal cooperation will be fostered and sustained. One salient feature of P2P streaming is that each peer is simultaneously a customer and a server. Another salient feature of P2P dynamic streaming is that there are multiple services, namely, the versions with different viewing qualities.

We study P2P dynamic streaming as a cooperative game, where peers watching the same version will form a cooperation set, and a peer's strategy therefore is not simply whether it should join a cooperation set, but instead which cooperation set to join and how much to contribute to the selected cooperation set. In our cooperative game, each peer bids with the maximum price he is willing to pay for the video and with his upload capacity. Based on peer bids, we use mechanism design to device pricing schemes for all video versions that reflect content and bandwidth cost savings derived from peer cooperation, as well as individual peer's upload contribution. Given each peer's bid and video prices at all versions, the mechanism determines the peers that can watch the highest-rate version while covering the server's cost (content and bandwidth) for providing the highest-rate video to the swarm. Typically a peer that is willing to contribute a large amount of upload bandwidth and is willing to pay a relatively high price will be assigned to the highest-rate swarm. Peers who are not assigned to the highest-rate swarm are considered for the next-highest rate swarm, and so on. Each peer is therefore assigned to a swarm that is commensurate with its upload contribution and the price it is willing to pay. Due to the cooperation gain on bandwidth and content price, the version to which a peer is assigned also depends on the version assignments for the other peers. We show that our pricing mechanisms can foster and sustain cooperation across different video versions. We also show that our pricing mechanisms are *group-strategy-proof*, i.e., peers are incentivized to bid their true upload capacities and their true maximum prices.

Using insights obtained from the game-theoretic analysis, we develop a multi-swarm dynamic P2P streaming system,

consisting of distributed P2P chunk scheduling algorithm, token-based accounting scheme that implements our video pricing schemes derived from cooperative game theory, and video version switching strategy that dynamically adjust each peer's video version based on the collaborative behaviors of all peers. Through chunk-level simulations, we demonstrate that our proposed incentive mechanisms can maximally incentivize heterogeneous peers to participate in P2P dynamic streaming.

This paper is organized as follows. We describe related work in the Section II. In Section III, we first briefly review cooperative game theory, we then apply cooperative game theory to single-version systems, and finally to multiple version systems. In Section IV, we describe our distributed dynamic P2P streaming algorithm, for which we provide simulation results in Section V. The paper is concluded in Section VI.

II. Related Work

Most research on dynamic adaptive streaming is based on the HTTP client/server architecture. Netflix [8] is currently one of the largest adaptive video streaming providers in the world. Its high-level system architecture is studied in [9] and more fine-grained architecture is studied in [10]. The performance of three popular HTTP adaptive streaming clients – the Netflix, Microsoft Smooth Streaming [11], and Adobe OSMF [12] clients – are compared in [3]. P2P streaming systems, such as PPLive [13] and PPStream [14], have been widely studied [15], [16], [17], [18], [19]. But none of them studied P2P dynamic streaming.

Incentive mechanism design is crucial to P2P systems. In [20], Chu et al. introduced taxation to P2P steaming to balance social welfare and individual welfare in the whole P2P community. In [21], Levin et al. proposed an auction solution to motivate users of BitTorrent [22] to share their upload bandwidth. In [23], Aperjis et al. developed a content pricing algorithm to encourage peers to share content in P2P system. There are also some previous work using game theory to study P2P incentives. In [24], Buragohain *et al* use game theory to study the interaction of strategic and rational peers, and propose a differential service-based incentive scheme to improve the system performance. In a more closely related work to this paper, [25], Misra *et al* use Shapely value to study the ideal incentive structure in P2P systems. They showed that, as the number of peers increases, the Shapley value received by each peer approaches a fluid limit, which can be obtained by a simple closed form expression. The book [26] systematically introduces game theory and related algorithms, and it has extensive coverage on cooperative games and cost sharing.

To the best of our knowledge, we are the first to use cooperative game theory to develop incentive mechanisms for P2P dynamic streaming.

III. Cooperative Games for P2P Streaming

A. Review of Cooperative Games and Cost-sharing

We begin with a short review of cooperative games and cost-sharing relevant to our study. (See [26] for a more detailed overview.) In a cooperative game, a set of agents are interested in receiving a service, but not all agents are necessarily provided the service. The total service cost is determined by the subset of agents that are provided the service, i.e, the cooperation (coalition) set. Cooperative game theory studies how to design cost-sharing mechanisms to encourage and sustain cooperation among agents.

Let $(\mathcal{A}, c)$ denote a cost-sharing game, where $\mathcal{A}$ is a set of agents interested in receiving a service, $c : 2^{\mathcal{A}} \rightarrow \mathcal{R}$ is the cost function of providing the service to any subset of agents. A *cost-sharing scheme* is a function that for each coalition $S \subseteq \mathcal{A}$, assigns a cost allocation for all agents in S. Formally, it is defined as follows:

Definition 1: A *Cost-sharing Scheme* is a function $\xi : \mathcal{A} \times 2^{\mathcal{A}} \rightarrow \mathcal{R}$ such that, for every $S \subseteq \mathcal{A}$, $\xi(i, S) = 0$ for all $i \notin S$, and $\xi(i, S) \in \mathcal{R}$ for all $i \in S$.[1] A cost-sharing scheme ξ is *budget balanced* if $\sum_{i \in S} \xi(i, S) = c(S)$ for all $S \subseteq \mathcal{A}$.

A cost sharing scheme determines, for any participating set of agents, the prices each agent has to pay. To determine the participating set, the service provider conducts an auction among the potential agents, and selects the agents based on the bids and the cost sharing scheme ξ. Specifically, let v_i be the utility of agent i for receiving the service, i.e., the maximum amount that agent i is willing to pay to receive the service. Suppose, for now, each agent i truthfully bids its utility v_i. A service set S is said to be *incentive-compatible* if $\{v_i \geq \xi(i, S), \forall i \in S\}$. After all the agents bid, the service provider simply determines all the incentive-compatible service sets, chooses one such set S (typically the largest), and assigns the cost sharing $\xi(i, S)$ to each agent i in S.

But in practice, without a properly designed cost-sharing mechanism ξ, a self-optimizing agent (or a coalition) may not bid truthfully in order to increase its profit $v_i 1(i \in S) - \xi(i, S)$, where $1(\cdot)$ is the indicator function. One of the goals of cooperative game theory is to design a balanced cost-sharing mechanism ξ that incentivizes all users to bid their actual utility values. To this end, we need the following definitions.

Definition 2: Given a cost-sharing game $(\mathcal{A}, c)$ and a vector $\mathbf{b} = \{b_i, i \in \mathcal{A}\}$ of service bids from all agents, a *cost-sharing mechanism* is an algorithm that determines a set $S(\mathbf{b}) \subseteq \mathcal{A}$ of agents to be serviced and a vector $\mathbf{p}(\mathbf{b}) = \{p_i, i \in \mathcal{A}\}$ of payments that the agents make to the service provider. $(S(\mathbf{b}), p(\mathbf{b}))$ is referred as the *output* of the cost-sharing mechanism.

Algorithm 1 Cost Sharing Mechanism $M_\xi(\mathcal{A}, \mathbf{b})$

1: Initialize $Q \leftarrow \mathcal{A}$.
2: Repeat
3: Let $Q \leftarrow \{i \in Q : b_i \geq \xi(i, Q)\}$.
4: Until for all $i \in Q, b_i \geq \xi(i, Q)$.
5: Return $S = Q$ and $p_i = \xi(i, Q)$, $\forall i \in Q$, $p_i = 0$, $\forall i \notin Q$

An example cost sharing mechanism, M_ξ, is given by Algorithm 1. Note that the mechanism is generated from a

[1] For generality, we don't require the cost allocation to be non-negative.

cost-sharing scheme ξ and a bidding vector $\mathbf{b}$ (which may not be truthful). A highly desirable property of a cost-sharing mechanism is that it provides incentives to agents to bid truthfully. This concept is formalized as follows:

Definition 3: Let $\mathbf{b} = \{b_i = v_i, \forall i \in \mathcal{A}\}$ be the truthful bid vector. Let $C \subseteq \mathcal{A}$ be a bidding coalition of agents, and $\mathbf{b}^{'}$ be a strategically chosen new bid vector, with $b_i^{'} = b_i, \forall i \notin C$. Let $(S, \mathbf{p})$ and $(S^{'}, \mathbf{p}^{'})$ denote the outputs of the mechanism when the bids are $\mathbf{b}$ and $\mathbf{b}^{'}$, respectively. A cost-sharing mechanism is *group-strategyproof* if for every coalition C of agents, if the inequality $b_i 1(i \in S') - p_i^{'} \geq b_i 1(i \in S) - p_i$ holds for every $i \in C$, then it holds with equality for every $i \in C$.

One of the principal goals in cooperative game theory is to find group-strategyproof cost-sharing mechanisms. Under a group-strategyproof mechanism, there is no benefit for agents (as well as groups of agents) to lie about their utilities.

Definition 4: A cost-sharing scheme ξ is *cross-monotone* if for all $S, T \subseteq \mathcal{A}$, $\xi(i, S) \geq \xi(i, S \cup T)$ for all $i \in S$.

Intuitively, a cross-monotone cost-sharing scheme provides incentives for agents to form larger cooperation sets to achieve lower per-agent charges. The following two theorems are classic results from cooperative game theory [26]:

Theorem 1: Suppose ξ is a cross-monotone cost sharing scheme for the cooperative game $(\mathcal{A}, c)$ and $\mathbf{b} = \{b_i, i \in \mathcal{A}\}$ is a bidding vector. Then there is a unique maximal service set $S^*(\mathbf{b}) \subseteq \mathcal{A}$ such that for any set S satisfying the property that $b_i \geq \xi(i, S)$ for all $i \in S$, we have $S \subseteq S^*(\mathbf{b})$. Furthermore, M_ξ in Algorithm 1 returns the maximal service set $S^*(\mathbf{b})$.

Theorem 2: If ξ is cross-monotonic cost-sharing scheme, then M_ξ is a *group-strategyproof cost sharing mechanism*.

The above two theorems indicate why the concept of cross-monotone cost-sharing scheme ξ and the mechanism M_ξ from Algorithm 1 are important. Specifically, if the cost-sharing scheme is cross monotone, under the mechanism M_ξ, every agent bids its his true utility value, $b_i = v_i$, and M_ξ returns the maximal service set $S^*(\mathbf{v})$ such that $v_i \geq \xi(i, S^*(\mathbf{v}))$ for all $i \in S^*(\mathbf{v})$.

B. Cost-sharing Mechanism for Single Version P2P Streaming

In P2P video streaming, cooperation between peers can give gains in both content and bandwidth. In this section, we apply cooperative game theory to study how to share the cost of a video streaming service among participating peers. One unique feature of this P2P streaming game is that a peer's participation in the service set can potentially reduce the system-wide bandwidth cost.

We start with the *basic P2P streaming cooperative game* where only one video version is available, and where each peer bids to see the video. We will extend this basic game to P2P streaming with multiple versions in the next subsection. We normalize the video rate to be 1 (and normalize the link bandwidths accordingly). We denote by $\mathcal{A}$ the set of peers interested in this video. Without loss of generality, we assume each peer in $\mathcal{A}$ has download bandwidth larger than the video rate. For peer i, its upload bandwidth is denoted by u_i and its utility for watching the video is v_i.

1) Service Cost: To serve a set S of peers, the server's cost can be calculated by

$$c(S) = c_c(S) + c_d(S), \tag{1}$$

where $c_c(S)$ is the total content cost and $c_d(S)$ is the total distribution cost for streaming the video to all peers in S. We assume the server content cost $c_c(S)$ is an increasing concave function of $|S|$. In other words, the server incurs decreasing marginal content cost for serving additional peers. Without P2P sharing, the distribution cost $c_d(S)$ is $w|S|$, where w is the cost per unit server bandwidth. When peer-assisted distribution is employed, video sharing between peers can significantly offload the server, thereby reducing the server distribution cost. In P2P streaming, the upload contribution of a peer is determined by its upload bandwidth as well as the efficiency of the P2P streaming algorithm in utilizing peers' upload bandwidth. As demonstrated in [27], if all peers are fully connected, all peers' upload capacity can be fully utilized to achieve high video streaming rate. In real P2P systems, however, a peer in a larger swarm has a better chance to find neighbors to exchange content, and it is also more likely to find nearby peers to achieve high P2P data-transfer throughput. As a result, the peer upload capacity utilization is less than 1 but normally increases as the number of peers increases [28], [18]. For our P2P streaming cooperative game, we model the average upload bandwidth utilization as an increasing concave function $\rho(|S|)$ of the streaming swarm size, with $\rho(|S|)$ approaching 1 when the number of peers is large. Taking into account the peers' upload contribution, the server's upload bandwidth contribution is given by:

$$U_d(S) = 1 + \left[|S| - 1 - \rho(|S|) \min(|S| - 1, \sum_{i \in S} u_i) \right], \tag{2}$$

where the first term is the unavoidable server bandwidth cost to stream at least one full copy of the video to the swarm, the second term is the additional bandwidth the server has to provide on top of peer upload contribution to ensure all peers receive the video stream.

2) Cost-sharing Mechanism for $u_i \leq 1$: When $u_i \leq 1$ for all $i \in \mathcal{A}$, so that each peer's upload bandwidth is no more than the video rate, server distribution cost can be approximated by

$$\hat{U}_d(S) = |S| - \rho(|S|) \sum_{i \in S} u_i = \sum_{i \in S} (1 - \rho(|S|) u_i), \tag{3}$$

with approximation error bound $-1 \leq \hat{U}_d(S) - U_d(S) \leq 0$. Using this minor approximation, the server cost becomes

$$c(S) = c_c(S) + w \sum_{i \in S} (1 - \rho(|S|) u_i).$$

Therefore, a natural cost-sharing scheme for peer i with upload capacity u_i is:

$$\textbf{Scheme-I:} \quad p(i, u_i, S) \triangleq \frac{c_c(|S|)}{|S|} + w(1 - \rho(|S|) u_i), \tag{4}$$

where the first term is the equal share of the content cost, the second term is the distribution bandwidth cost discounted by each peer's upload contribution.

Theorem 3: **Scheme-I** is *cross-monotone.*

Proof: For all $S, T \subseteq N$ and $i \in S$, due to the concavity of $c_c(\cdot)$, $\frac{c_c(S)}{|S|} \geq \frac{c_c(S \cup T)}{|S \cup T|}$. Also since $\rho(\cdot)$ is an increasing function, we have $\rho(|S|) \leq \rho(|S \cup T|)$. Therefore, we have $p(i, u_i, S) \geq p(i, u_i, S \cup T)$, which satisfies the definition of *cross-monotone.* ■

As part of the bidding, if each peer reveals its true upload bandwidth (but not necessarily its true utility), then we can directly apply the theory in Section III-A. In particular, it follows from Theorem 3, and Theorem 1 and 2 in Section III-A that under the cost-sharing **Scheme-I** and the mechanism of Algorithm 1, every peer will bid its true utility value, v_i, and Algorithm 1 returns the maximal service set $S^*(\mathbf{v})$.

But it is more natural to suppose that each peer may not only lie about its utility v_i but also lie about its upload bandwidth u_i. In particular a peer (or a group of colluding peers) may bid a higher upload bandwidth than it actually has in order to increase its chances of being included in the swarm. In this case, the bidding game is different from the traditional game reviewed in Section III-A. Recall that in the traditional game, the total cost $c(S)$ and the cost share for each agent $\xi(i, S)$ is determined only by the cooperation set S and not the bids. But now, since the cost shares $p(i, u_i, S)$ depend on the bids (the upload rates), the classical cooperative game theory no longer directly applies. Algorithm 2 develops a cost-sharing mechanism $M(p)$ out of a cost-sharing scheme p for a P2P cooperative game.

Algorithm 2 P2P Cost Sharing Mechanism $M(p)$

1: Each peer bids its utility and upload bandwidth $\langle v_i, u_i \rangle$;
2: Initialize $Q \leftarrow \mathcal{A}$.
3: Repeat
4: Let $Q \leftarrow \{i \in Q : v_i \geq p(i, u_i, Q)\}$.
5: Until for all $i \in Q, v_i \geq p(i, u_i, Q)$.
6: Return $S = Q$ and $p_i = p(i, u_i, Q)$, for all $i \in Q$, $p_i = 0$, for all $i \notin Q$.

To summarize the P2P cooperative game, the following steps are taken:

1) We design a cost structure $p(i, \mathbf{u}, S)$ which in general may depend on the vector of upload bandwidths $\mathbf{u} = \{u_i, i \in \mathcal{A}\}$.
2) Each peer i bids $\langle v_i', u_i' \rangle$, which may or may not be truthful.
3) After the bidding is completed, Algorithm 2 is used to determine the set of participants S' based on the $\langle v_i', u_i' \rangle$ values.
4) The P2P streaming is carried out over the set S'. Peer i is charged according to his actual upload contribution u_i rather than his bidded upload capacity u_i'. The profit for peer i is given by $v_i - p(i, \mathbf{u}, S')$ when $i \in S'$ and is zero otherwise.

Theorem 4: In a P2P cooperative game, if the cost-sharing scheme p is cross-monotone and the cost sharing of each peer $p(i, \mathbf{u}, S)$ only depends on $\mathbf{u}$ through its own upload contribution u_i, then the cost-sharing mechanism $M(p)$ defined in Algorithm 2 is *group-strategyproof.*

Proof: Let S be the service set returned by $M(p)$ under truthful bids. Since p is a cross-monotone cost-sharing scheme, S is the unique maximal set satisfying $v_i - p(i, u_i, S) \geq 0$, $\forall i \in S$.

We prove by contradiction. Specifically, suppose there is a group of colluding peers C and associated bids $\langle v_i', u_i' \rangle$, $i \in C$, resulting in service set S' returned by $M(p)$, with the following properties: (a) $v_i 1(i \in S') - p(i, u_i, S') \geq v_i 1(i \in S) - p(i, u_i, S)$ holds for every $i \in C$, and (b) there is at least one $i_0 \in C$ such that $v_{i_0} 1(i_0 \in S') - p(i_0, u_{i_0}, S') > v_{i_0} 1(i_0 \in S) - p(i_0, u_{i_0}, S)$.

Consider a peer $i \in S' - C$. Such a peer bids truthfully, and according to the definition of $M(p)$, we immediately have $v_i - p(i, u_i, S') \geq 0$. Now consider a colluding peer $i \in S' \cap C$. By (a) we have $v_i - p(i, u_i, S') \geq v_i 1(i \in S) - p(i, u_i, S)$. If $i \notin S$, then it directly follows $v_i - p(i, u_i, S') \geq 0$. If $i \in S$ then it follows $v_i - p(i, u_i, S') \geq v_i - p(i, u_i, S) \geq 0$, where this last inequality follows from the property of S stated in the first paragraph of this proof. Combining these cases, we have $v_i - p(i, u_i, S') \geq 0$, $\forall i \in S'$. Since S is the maximal service set under truthful bid, then $S' \subseteq S$.

For (b) to hold, it is easy to see that $i \in S'$ and thus $i \in S$. Thus, $v_{i_0} - p(i_0, u_{i_0}, S') > v_{i_0} - p(i_0, u_{i_0}, S)$. Then $p(i_0, u_{i_0}, S') < p(i_0, u_{i_0}, S)$, which contradicts $S' \subseteq S$ and p is cross-monotone. Thus, $M(p)$ is group-strategyproof. ■

Corollary 4.1: M(Scheme-I) is group-strategyproof.

3) Cost-sharing Scheme for the General Case: For the case where there is at least one $i \in \mathcal{A}$, with $u_i > 1$, the server cost approximation in (3) can introduce considerable error for some peer combinations. For this case, we need to design a new cost-sharing scheme. Motivated by Scheme-I, it is natural to consider **Scheme-II** to reward a peer by its relative upload contribution:

$$p(i, u_i, S) \triangleq \frac{c_c(S)}{|S|} + w\left(1 - \min(1, \frac{|S| - 1}{\sum_{i \in S} u_i}) u_i \rho(|S|)\right),$$

where the first term is equal content cost share, and the second term is the adjusted peer upload contribution. Note that in Scheme-II, the costs $p(i, \mathbf{u}, S)$ now depend on $\mathbf{u}$ and not just on u_i. It is straight-forward to establish the following result:

Theorem 5: Scheme-II is *budget-balanced.*

Although Scheme-II is very natural and budget-balanced, it is neither cross-monotone nor group-strategyproof.

Theorem 6: Scheme-II is not cross-monotone.

Proof: We provide a counter example. Suppose we have four peers, with upload bandwidth of $0, 0, 2, 2$ respectively. When only the first three peers form a coalition, $S_3 =$

$\{1,2,3\}$, the price for peer 3 is:

$$\begin{aligned} p(3,2,S_3) &= \frac{c_c(S_3)}{|S_3|} + w(1-\min(1,\frac{|S_3|-1}{\sum_{i\in S_3} u_i})u_i\rho(|S_3|)) \\ &= \frac{c_c(S_3)}{3} + w(1-\min(1,\frac{2}{2})2\rho(3)) \\ &= \frac{c_c(S_3)}{3} + w(1-2\rho(3)) \end{aligned}$$

When all four peers form a coalition, $S_4 = \{1,2,3,4\}$, the price for peer 3 is:

$$\begin{aligned} p(3,2,S_4) &= \frac{c_c(S_4)}{|S_4|} + w(1-\min(1,\frac{|S_4|-1}{\sum_{i\in S_4} u_i})u_i\rho(|S_4|)) \\ &= \frac{c_c(S_4)}{4} + w(1-\min(1,\frac{3}{4})2\rho(4)) \\ &= \frac{c_c(S_4)}{4} + w(1-\frac{3}{2}\rho(4)) \end{aligned}$$

If $\frac{c_c(S_3)}{3} - \frac{c_c(S_4)}{4} < \frac{w\rho(3)}{2}$, $p(3,2,S_3) < p(3,2,S_4)$, establishing that Scheme-II is not cross-monotone. ∎

Theorem 7: Let M(Scheme-II) be the mechanism obtained from Algorithm 2. M(Scheme-II) is not *group-strategyproof.*

Proof: We prove this using the example in the proof of Theorem 6. Set the utility values large enough for all four peers so that if they all bid truthfully, they will all be included in the service set, that is, set $v_i \geq p(i,u_i,S_4), i = 1,2,3,4$. The cost shares for peer 3 and 4 are the same, as calculated in (5):

$$p(3,2,S_4) = p(4,2,S_4) = \frac{c_c(S_4)}{4} + w(1-\frac{3}{2}\rho(4)).$$

Now suppose peer 3 announces that its upload bandwidth is 10 instead of 2. The cost shares for peers 3 and 4 become

$$\begin{aligned} p'(3,10,S_4) &= \frac{c_c(S_4)}{4} + w(1-\min(1,\frac{2}{12})10\rho(4)) \\ &= \frac{c_c(S_4)}{4} + w(1-\frac{5}{3}\rho(4)) \qquad (5) \end{aligned}$$

and

$$\begin{aligned} p'(4,2,S_4) &= \frac{c_c(S_4)}{4} + w(1-\min(1,\frac{2}{12})2\rho(4)) \\ &= \frac{c_c(S_4)}{4} + w(1-\frac{1}{3}\rho(4)) \qquad (6) \end{aligned}$$

respectively. Thus, if we set peer 4's utility value to $v_4 = \frac{c_c(S_4)}{4} + w(1-\frac{1}{2}\rho(4)) < p'(4,2,S_4)$, and set the other three peers to have large values, then the resulting coalition set for $M(p')$ becomes $\{1,2,3\}$. Since $u_1 = u_2 = 0$, it is easy to check that the cost share allocated to peer 3 is $p'(3,10,S_3) = p(3,2,S_3) < p(3,2,S_4)$ as shown in Theorem 6. Thus, because peer 3 gains benefit by lying about its upload bandwidth, M(Scheme-II) is not *group-strategyproof.* ∎

Based on the above cost-sharing scheme, now consider **Scheme-III**:

$$p(i,u_i,S) = \frac{c_c(S)}{|S|} + w\left(1-\min(1,\frac{|S|-1}{|S|\bar{u}})u_i\rho(|S|)\right),$$

where $\bar{u}$ denotes the average upload bandwidth of peers who are watching the video. The value can be determined by the server using historical data. It is straightforward to show that Scheme-III is *budget-balanced* if $\bar{u}$.

Theorem 8: Scheme-III is *cross-monotone.*

Proof: For all $S,T \subseteq \mathcal{A}$ and $i \in S$, we have $p(i,u_i,S) = \frac{c_c(S)}{|S|} + w(1-\min(1,\frac{|S|-1}{|S|\bar{u}})u_i\rho(|S|))$, $p(i,S\cup T) = \frac{c_c(S\cup T)}{|S\cup T|} + w(1-\min(1,\frac{|S\cup T|-1}{|S\cup T|\bar{u}})u_i\rho(|S\cup T|))$. Since $c_c(\cdot)$ is a concave function, we have $\frac{c_c(S)}{|S|} \geq \frac{c_c(S\cup T)}{|S\cup T|}$. And since $\rho(\cdot)$ is an increasing function, we have $\frac{|S|-1}{|S|\bar{u}}\rho(|S|) \leq \frac{|S\cup T|-1}{|S\cup T|\bar{u}}\rho(|S\cup T|)$. Thus, we have $p(i,u_i,S) \leq p(i,u_i,S\cup T)$, which is the definition of *cross-monotone.* ∎

Together with Theorem 1 and 2, we immediately have

Corollary 8.1: Let M(Scheme-III) be the mechanism obtained from Algorithm 2. M(Scheme-III) is group strategyproof.

C. Multi-Version P2P Streaming

Now suppose there are J video versions with decreasing video rates $r_1 > r_2 > \cdots > r_J$. Similar to the single-version case, there is a set $\mathcal{A}$ of peers. Peer i has upload capacity of u_i, and video utility v_i. We assume each peer i first and foremost wants to watch the highest video rate allowed by its utility budget v_i. After being assigned to the highest possible video rate, it desires to pay the lowest possible price without changing the assigned rate.

Let S_j be the set of peers watching version j. We set the pricing scheme for peer i watching video version j based on **Scheme-III:**

$$p_j(i,u_i,S_j) = c_j(|S_j|) + w(r_j - \min(1,\frac{(|S_j|-1)r_j}{|S_i|\bar{u}_j})u_i\rho(|S_j|)), \qquad (7)$$

where $c_j(|S_j|) \triangleq c_c^{(j)}(|S_j|)/|S_j|$ is the content price for version j, $\bar{u}_j$ is the expected upload bandwidth of peers watching version j.

Algorithm 3 Top-Down Multi-Round Bidding for Multi-version P2P Streaming

```
Each peer bids its utility and upload bandwidth ⟨v_i, u_i⟩;
Initialize S ← A.
for j = 1, ..., J do
    T = S
    REPEAT
        Let S ← {i ∈ S : v_i ≥ p_j(i, u_i, S)}.
    Until for all i ∈ S, v_i ≥ p_j(i, u_i, S).
    S_j = S; R_i = r_j, for all i ∈ S_j;
    S = T − S_j;
    If S = ∅ then Stop.
end for
If S ≠ ∅ then R_i = 0 for all i ∈ S;
```

Algorithm 3 selects video rate R_i for each peer i. It starts from the highest video rate. Once a peer is selected to watch a particular video rate, it is not considered for the lower video rates. When each peer has been allocated to some video version, or all video versions have been considered, the algorithm ends.

Theorem 9: Top-down bidding in Algorithm 3 is *group-strategyproof.*

Proof: Given peer bids, Algorithm 3 determines the video version for each peer. Let $S_j, 1 \leq j \leq J$, be the set of peers assigned to video version j if all peers bid truthfully. Suppose a set C of peers collude in bidding. Peer $i \in C$ bids with $\langle v_i', u_i' \rangle \neq \langle v_i, u_i \rangle$. Let $S_j', 1 \leq j \leq J$, be the new peer video rate assignments returned by the top-down bidding algorithm.

Start with video version $j = 1$. Since $p_1(i, u_i, S_1)$ satisfies the condition of Theorem 4, using the same argument in the proof of Theorem 4, it can be shown that $S_1' \subseteq S_1$.

1) If some colluders are allocated to the highest video version in the truthful bidding case, i.e. if $C \cap S_1 \neq \emptyset$, since colluders cannot suffer loss, those colluders should still be allocated to the highest version, i.e., $C \cap S_1 \subseteq S_1'$. Additionally, they should not be charged with a higher price than in the truthful bid case, i.e., $p_1(i, u_i, S_1') \leq p_1(i, u_i, S_1)$. Due to the cross-monotonicity of $p_1(\cdot)$, we must have $S_1' = S_1$.

2) If no colluder is allocated to the highest video version in the truthful bidding case, i.e. $C \cap S_1 = \emptyset$, even under group collusion, all peers in S_1 still bid truthfully, and they all will survive in the first round of bidding for the highest video rate, i.e, $S_1 \subset S_1'$. Combining with $S_1' \subseteq S_1$, we also have $S_1' = S_1$.

Summarizing the previous two cases, we will always have $S_1' = S_1$. According to the top-down bidding process, the set of peers participating in bidding for video rate $j = 2$ is the same as the set under truthful bidding. The previous arguments can be applied similarly for second-round bidding game to show that $S_2' = S_2$. Repeating the arguments for all video versions, we can show that $S_j' = S_j$, $1 \leq j \leq J$. Then each colluder will be allocated to the same video rate and charged with the same price. Therefore, no peer can benefit from collusion. Thus, the top-down bidding mechanism in Algorithm 3 is *group-strategyproof.* ■

IV. Dynamic P2P Streaming Algorithm

Now we use the cost-sharing mechanisms developed in the previous section to guide the video rate selection of peers in dynamic P2P streaming. We develop distributed dynamic P2P streaming algorithms – consisting of chunk scheduling, token-based accounting, and video version switching – to dynamically adjust peers' video rates based on collaborative behaviors of all peers.

A. Multi-swarm Chunk Scheduling

Each version of the video is divided into video chunks, each of which has video playback time of Δ. All peers watching the same version form a P2P streaming swarm and exchange video chunks. When a peer decides to switch from version j to version j', it simply leaves the swarm j and joins swarm j'. For each swarm, we employ a *mesh-pull* based P2P streaming design [16]. In such a design, a tracker keeps track of all the peers in the swarms. A newly joined peer receives an initial list of peers from the tracker, and randomly connects to a subset of them. Additional peer lists can be obtained from connected neighbors. Peers obtain video chunks from their neighbors or the server. Each peer maintains a sliding window of chunks to download, moving forward at the video playback rate. Peers exchange with their neighbors their buffer-maps, indicating which chunks are available in their local cache buffers.

For chunk download scheduling, a peer employs a hybrid of *oldest-first* and *rarest-first* strategies [29]: 1) a chunk close to its playback deadline is given high download priority; 2) among chunks far away from their deadlines, chunks that are less available among neighbors are given higher download priority to create chunk diversity among neighbors for P2P sharing. To maximally offload the server, a peer first requests a chunk from its neighbors if it is available. When a chunk is available from multiple neighbors, to give all peers the same chance to contribute their upload bandwidth, as assumed in the pricing mechanism (7), the request will be sent to the neighbor with the lowest upload bandwidth utilization. This also balances neighbor workloads, and consequently chunk download delays. If all neighbors are overloaded, the peer either downloads the chunk from the server in this round, or waits and downloads from neighbors in future rounds. If the chunk is very close to its playback deadline, it will be downloaded directly from the server; otherwise it will be downloaded from the server in the current round with a certain probability, calculated from a decreasing function of the chunk availability in its neighbors. For chunk upload scheduling, we employ a simple *First-Come-First-Serve* policy at the peers and at the server.

B. Token-based Accounting

We implement the cost-sharing mechanisms proposed in Section III using *tokens*, which are exchanged among the server and peers. Each peer buys tokens from the server periodically at the rate determined by its video utility value. Specifically, if the video duration is T, and peer i's utility value for the video is v_i, then peer i buys tokens at rate v_i/T. To facilitate chunk-based P2P video streaming and video switching, we employ *chunk-based mini-payments*: when a peer downloads a chunk, it pays the distribution cost to the uploader (either the server or another peer), and the content cost to the server. The content and distribution costs are calculated using the cost-sharing schemes designed in Section III.

For the clarity of presentation, suppose time is slotted with unit time interval, and peers only switch video versions at the end of each time slot. At the beginning of time slot k, for peer i, let $B_i(k)$ be its token balance, $j_i(k)$ its current video version, and $r_{j_i(k)}$ the corresponding video rate. Let $D_i(k)$ and $U_i(k)$ be the total number of chunks downloaded and uploaded, respectively, by peer i within time slot k. Let $S_j(k)$ be the set of peers watching video version j in time slot k. The token balance on peer i is updated as:

$$B_i(k+1) = \max\left\{0, B_i(K) + \frac{v_i}{T} + U_i(k)\frac{wr_{j_i(k)}\Delta}{T} - D_i(k)\left(\frac{wr_{j_i(k)}\Delta}{T} + \frac{c_{j_i(k)}(|S_{j_i(k)}(k)|)\Delta}{T}\right)\right\}, \tag{8}$$

where the v_i/T is the number of tokens bought by peer i according to utility v_i, $wr_{j_i(k)}\Delta/T$ is per-chunk distribution cost for version $j_i(k)$, $U_i(k)wr_{j_i(k)}\Delta/T$ is the number of tokens earned by peer i by uploading $U_i(k)$ chunks of version $j_i(k)$ to other peers, and the last term is the number of tokens paid by peer i for downloading $D_i(k)$ chunks, with $c_{j_i(k)}(|S_{j_i(k)}(k)|)\Delta/T$ being the per-peer per-chunk content cost of version $j_i(k)$ shared between $|S_{j_i(k)}(k)|$ peers.

To maintain smooth playback, we will have $D_i(k) = 1/\Delta$, the average upload rate of peer i within time slot k is $\tilde{u}(k) = U_i(k)r_{j_i(k)}\Delta$. To sustain token balance growth while continuously watching version j, peer i should have

$$\frac{v_i}{T} + \frac{w\tilde{u}(k)}{T} \geq \frac{wr_{j_i(k)}}{T} + \frac{c_{j_i(k)}(|S_{j_i(k)}(k)|)}{T}.$$

The left part is the rate peer i earns tokens, and the right part is the rate peer i consumes tokens. Equivalently, we have

$$v_i \geq c_{j_i(k)}(|S_{j_i(k)}(k)|) + w(r_{j_i(k)} - \tilde{u}(k)). \tag{9}$$

Compared with (7), the right part of (9) is the cost-share of peer i for version $j_i(k)$, given its real upload contribution of $\tilde{u}(k)$.

C. Dynamic Video Rate Switching

Based on the token balance, and the rate tokens are earned and consumed, peers make their video switching decisions.

1) Switching-Up: If a peer's token balance is high, then its current video rate is likely lower than what it can afford, so the peer should consider switching-up to a higher rate. Before switching-up to swarm j, the peer estimates whether it will be able to afford to watch at a higher rate in swarm j. We assume a peer can obtain the current content price $c_j(|S_j(k)|)$ of version j from the tracker. For a peer to estimate its upload contribution if it switches to version j, the tracker also provides the peer the upload bandwidth utilizations $\rho(S_j(k))$ in the target swarm j. Peer i then calculates its net token growth rate if it were to switch to version j:

$$I(i, j, S_j(k)) = \frac{v_i + w(u_i \cdot \rho(S_j(k)) - r_j) - c_j(|S_j(k)|)}{T} \tag{10}$$

If $I(i, j, S_j(k)) > 0$, peer i is expected to have token balance growth after switching to version j. Even if $I(i, j, S_j(k)) < 0$, with current token balance $B_i(k)$, peer i can switch to version j for a time period of $B_i(k)/|I(i, j, S_j(k))|$. Peer i can switch to the highest version j^* such that $B_i(k) \geq -\eta I(i, j, S_j(k))$, where η is the duration threshold to avoid frequent switches.

2) Switching-Down: A peer has to switch down to a lower version if its token balance hits zero. Similar to the switch-up case, with information from the tracker, a peer can calculate its net token growth rate at any lower video version using (10), and switch down to the highest version j^* with $I(i, j, S_j(k)) \geq 0$.

V. Performance Evaluation

To evaluate the performance of the proposed dynamic P2P streaming mechanisms and algorithms, we implemented an event-driven P2P streaming simulator based on the source C++ code provided by [30]. The simulator can simulate chunk-level P2P streaming and end-to-end latency among end peers. We extended it to support multiple streaming rates and implemented our chunk scheduling, token-based accounting and video version switching algorithms. As commonly assumed in P2P system study, we simulate an environment where the peer uplinks are the only bandwidth bottlenecks. The simulation has totally 200 peers, with one node being the server and the remaining 199 nodes being the peers. We created six different classes of peers according to Table I. The server has no upload bandwidth limit.

TABLE I: Peer Classes based on Utility and Bandwidth

Class Index	Bandwidth(Mbps)	Utility	Node Count
0	5.0	1.2	30
1	5.0	0.9	30
2	3.0	0.8	40
3	3.0	0.6	40
4	1.0	0.6	30
5	1.0	0.3	29

There are totally 6 different video rates which are $0.5Mbps, 1.5Mbps, 2.5Mbps, 3.5Mbps, 4.5Mbps, 5.5Mbps$. We run each simulation for $10,000$ seconds. We set the playback time of a chunk to 200 msec and the initial startup buffer delay to 4 seconds, i.e., the playback starts when the peer has obtained 20 consecutive chunks. During playback, if one chunk is missing, the current playback lag determines the playback freezing duration before moving to the next chunk. In our implementation, if the current playback lag is smaller than 6 seconds, the maximum freezing time is 2 seconds; if the current playback lag is between 6 seconds and 20 seconds, the maximum freezing time is 0.6 second; if the accumulated playback lag is more than 30 seconds, missing chunks will be skipped immediately.

A. P2P Chunk Scheduling Efficiency

We first demonstrate that our chunk scheduling algorithm is efficient for video streaming. For this purpose, we disable token accounting and force all peers to watch the same version at rate $4.5Mbps$. Fig.1 shows CDF of the chunk download delays for peers at different upload bandwidth levels. Fig.1 shows that all chunks can be downloaded well before the initial playback delay of 4 seconds. No playback freezing is observed. Table II shows the average peer upload rate, peer upload capacity utilization, the download ratio from the server, the percentage of chunks downloaded directly from the server, and the average playback delay. We can see that the chunk playback delay is just a little longer than pre-set initial startup delay of 4 seconds. We can see from Table II that our chunk scheduling algorithm can achieve close to 100% peer upload capacity utilization. Since the average peer upload capacity is $3Mbps$, about 36% percent of the $4.5Mbps$ video stream has to be downloaded from the server. This set of simulations demonstrates that our chunk scheduling algorithm is very

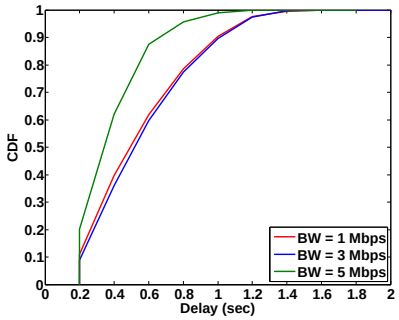


Fig. 1: Chunk Download Delay CDF without Switching

TABLE II: Chunk Scheduling Efficiency without Switching

Capacity Level	Upload Rate (Mbps)	Upload Utiliz. (%)	Download from Server (%)	Playback Delay(sec)
0	4.99	99.8	44.2	4.34
1	2.74	91.3	27.6	4.57
2	0.91	91.0	39.8	4.50

efficient in utilizing peer upload capacity for high quality live streaming.

B. Dynamic Streaming with Content Cost

To test the token-based accounting and dynamic video switching, we first conduct simulations with both content and distribution charges. The bandwidth price is set to $w = 1$, and the content price for video version j is set as $c_j(|S_j|) = 2 * r_j * \zeta(|S_j|)$, where $\zeta(\cdot)$ is a piece-wise constant group discount function of the group size, as defined in Table III. When more peers watch the same version, each of them will get a lower content price.

TABLE III: Group Content Discount Factor

Group Size	$1 \sim 9$	$10 \sim 19$	$20 \sim 29$	$30 \sim 39$	$40 \sim 199$
Per-peer Price	1	0.9	0.8	0.7	0.6

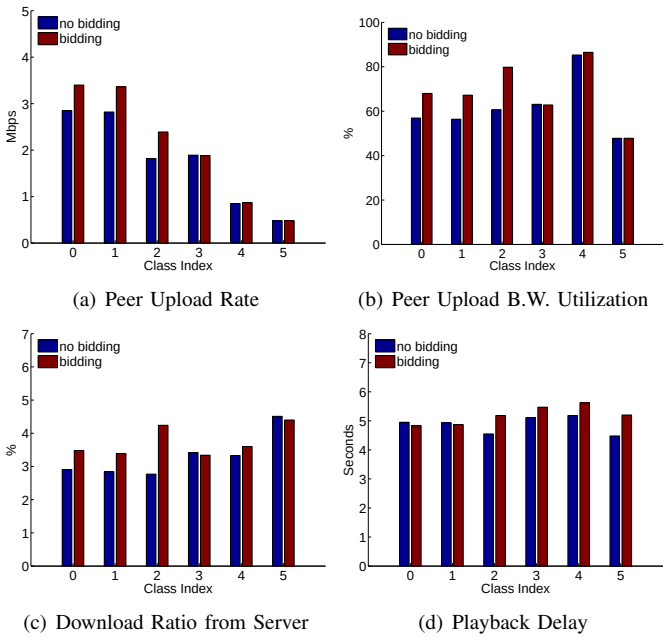
We study peer dynamic video switching with and without going through the top-down bidding. Without bidding, each peer starts with the lowest video rate of $0.5Mbps$, and dynamically switches up to higher rates as its token balance grows. With bidding, each peer participates in the top-down bidding and obtains its initial video rate assignment. We use the average peer upload bandwidth of $3Mbps$ as $\bar{u}$. A peer initially joins the video swarm for the assigned rate. After joining, the peer adjusts its video rate dynamically based on its token balance. In our simulations, both with and without bidding, video rates on peers converge quickly. Table IV compares the distributions of the converged video rates for different classes of peers in both cases.

We can see from Table IV, both with and without bidding, peers in class 5 get stuck at the lowest rate, due to their low utility and bandwidth values; peers in class 4, since they have a higher utility, accumulate tokens and are able to watch at rate $1.5Mbps$; peers in class 3 also watch at rate $1.5Mbps$; peers in class 2 watch at $2.5Mbps$. We can see that without bidding, all

TABLE IV: Video Rate Distribution of Different Classes

Class Index	0	1	2	3	4	5
Average Video Rate(Mbps) Start with Lowest Rate	2.5	2.5	2.5	1.5	1.5	0.5
Average Video Rate(Mbps) Start with Bidding Rate	3.5	3.5	2.5	1.5	1.5	0.5

peers in class 0 and 1 can increase their rates only to $2.5Mbps$, which is lower than their assigned rate of $3.5Mbps$, calculated by the top-down bidding algorithm. Meanwhile, if the peers instead start with their bidding-assigned rates, they can watch $3.5Mbps$ throughout the whole simulation, as indicated in Table IV. This is because when each peer starts with the lowest rate and switches up independently, they cannot coordinate to get the group discount in Table III, then they don't have enough balance to switch up to higher rate. Specifically, for the first peer attempting to switch to a high rate swarm, when it predicts its token growth rate after switching using (9), there is no group discount since there is no other peer in that swarm. The total predicted price might be higher than its utility value, in which case it decides to stay at the current rate. This applies to the subsequent attempts as well. As a result, no peer will jump up to that swarm. On the other hand, if they coordinate and synchronously jump up to a higher swarm, the group discount will allow them to maintain a positive token growth rate and watch high quality video. The initial top-down bidding therefore provides a means for peers to cooperate and collaboratively improve their video experience.



(a) Peer Upload Rate (b) Peer Upload B.W. Utilization

(c) Download Ratio from Server (d) Playback Delay

Fig. 2: Performance with/without Initial Bidding

Fig 2 compares the average peer upload bandwidth utilization, download ratio from the server, and playback delay of each class for without and with bidding. From Fig 2(a) and 2(b), we can see that for peers in class 0, 1 and 2,

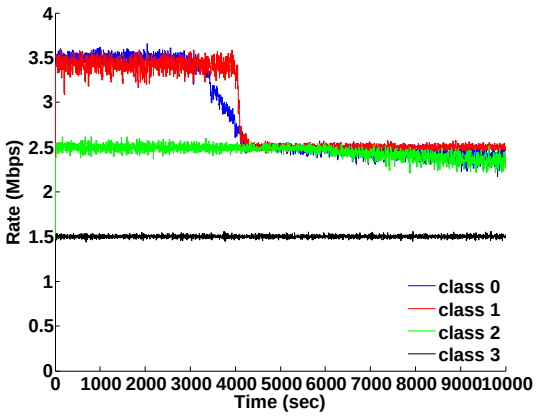


Fig. 3: Domino Effect of Peer Utility Change

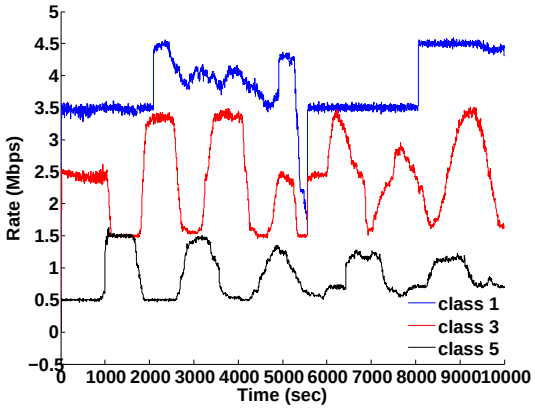
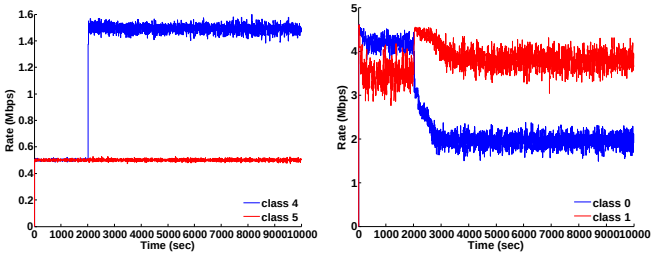


Fig. 4: Oscillations in Aggressive Switch-ups



(a) $1Mbps$ peers (b) $5Mbps$ peers

Fig. 5: Video Rate Switching Triggered by Upload Bandwidth Changes

their bandwidth utilization can get large improvement with initial bidding. For peers in class 3, 4 and 5, because their bandwidth is small, they don't get much improvement on their bandwidth utilization. Fig 2(c) shows that the download ratio from the ratio becomes a little bit larger with initial bidding, this happens because average video rate of class index 0, 1 and 2 with initial bidding is higher than that of without initial bidding, as shown in Table IV. Also, we can see from Fig 2(d) that playback delay is still short after applying initial bidding. The server upload bandwidth is mostly used to stream chunks close to their deadlines. The streaming performance is stable across the peers watching different rates. Due to the content pricing, peers cannot watch video rates matching their upload capacities. This leads to peer upload bandwidth utilization that is lower than 100%.

To test the adaptivity of our system, we change the utility value of class 0 from 1.2 to 0.3 at $2,000$ seconds into the simulation. Fig.3 shows the video download rate fluctuations after that. There is no change for peers in class 5 and 4; Fig.3 only shows the average video download rates of peers in class $0, 1, 2, 3$.

The average video rate of class 0 starts to drop at about 500 seconds after the change. It doesn't drop immediately because it takes some time for the peers to deplete their token balances. As class 0 peers leave the $3.5Mbps$ swarm, the content price for version $3.5Mbps$ increases according to Table III. The token balance for class 1 peers then starts to decrease. The content price increases at faster pace as more peers leave the swarm, leading to a domino effect for switching down for peers in class 1 at approximately $4,000$ seconds into the simulation. Interestingly, the average video rate of class 2 also decreases slightly towards the end of the simulation. As more class 0 and class 1 peers switch down to $2.5Mbps$, more chunk download requests will go to them instead of to class 2 peers. Because some class 2 peers cannot earn enough tokens from chunk uploading, they will be forced to switch down to a lower rate. Class 3 peers are not influenced, because the $1.5Mbps$ swarm is not affected by class-0 utility changes.

For the above simulations, a peer switches up to video rate j only if the expected token growth rate $I(i, j, S_j(k))$, calculated by (10), is positive. As discussed in Section IV-C, a peer can also aggressively switch up to level j even if the expected token growth rate is negative, but its balance allows it to stay there for η time, i.e., $B_i(k) \geq -\eta I(i, j, S_j(k))$. If the predicted growth rate is correct, the peer will have to switch down when the token balance depletes. We conducted simulations for the aggressive switch up strategy with $\eta = 10$ minutes. As shown in Fig.4, as expected, aggressively switching-up triggers many video rate oscillations. And also, we can see that the period of the oscillation is about 10 minutes as expected. This is acceptable when the remaining video time is less than η because even the token growth rate is negative, the existing balance can afford video time of η at a high rate.

C. Dynamic Streaming with Free Content

When content is free, the video rate watched by a peer is mainly determined by its own utility value and its real upload contribution. We now reduce peer utility values as shown in Table V. In this case, the token balance growth rate is largely dependent on a peer's real upload contribution. Both with and without bidding, the dynamic video rate adjustment algorithm will drive peers to swarms corresponding to their utility values and upload contributions. This is because there is no group discount factor for content any more. Due to space limit, we only present results for dynamic rate switching triggered

TABLE V: Settings for Content-Free Simulation

Class Index	Bandwidth(Mbps)	Utility	Node Count
0	5.0	0.1	30
1	5.0	0.07	30
2	3.0	0.1	40
3	3.0	0.07	40
4	1.0	0.1	30
5	1.0	0.07	29

by upload bandwidth changes. We conduct simulations by changing the bandwidth of class 0 to $2.5Mbps$ and class 4 to $2Mbps$ at 2000 seconds into the simulation The video download rates of all classes are plotted in Fig.5.

We can see that class 4 peers switch up from $0.5Mbps$ to $1.5Mbps$ immediately, and the video rate of class 0 peers drops down after some time. This is because it takes some time to deplete the token balances. Some class 1 peers also switch up, since they have a better chance to earn tokens with some class 0 peers switching down. But as more class 0 peers switch down, the swarm becomes smaller and many previously switched-up class 1 peers switch back. The peers from classes 2 and 3 are almost unaffected.

VI. Conclusion

Dynamic video streaming is a new trend on the Internet. While all existing dynamic streaming solutions are server-based, P2P dynamic streaming holds a great potential. The success of P2P dynamic streaming hinges on incentive mechanism design. In this paper, we use cooperative game theory to study the incentive issues in P2P dynamic streaming. We developed cost-sharing mechanisms for peers watching the same video version. We also developed a top-down multi-round bidding mechanism for heterogeneous peers to select video versions that are commensurate with their upload contributions and the prices they are willing to pay. We showed that our cost-sharing and bidding mechanisms are *budget-balanced* and *group-strategyproof*. Using insights obtained from the game-theoretic study, we developed a P2P dynamic streaming system, consisting of multi-swarm chunk scheduling, token-based accounting, and video version switching, to dynamically adjust each peer's video version based on the collaborative behavior of all peers. Through simulations, we demonstrated the efficiency of the proposed P2P incentive mechanism and dynamic streaming algorithm.

As future work, we will refine our P2P dynamic streaming system design. We will improve the robustness of the current streaming algorithm within each video swarm as peers dynamically switch between versions. We will further investigate the trade-off between the smoothness and responsiveness of peer video version switching, leveraging on our recent study on server-based dynamic streaming [31]. Finally, we plan to develop a P2P dynamic streaming prototype and test it through experiments over the Internet.

References

[1] J. F. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, 6th ed. Addison-Wesley, 2012.

[2] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proceedings of ACM conference on Multimedia systems*, 2011.

[3] ——, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," 2011.

[4] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth?" in *NOSSDAV*, 2012.

[5] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," in *Proceedings of IEEE INFOCOM*, 2012.

[6] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *ACM Internet Measurement Conference*, 2012.

[7] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2012.

[8] Netflix, "Netflix Homepage," http://www.netflix.com.

[9] M. Watson, "Http adaptive streaming in practice," Netflix, Tech. Rep., 2011.

[10] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," in *Proceedings of IEEE INFOCOM*, 2012.

[11] Microsoft, "IIS Smooth Streaming," http://www.iis.net/download/SmoothStreaming.

[12] Adobe, "Open Source Media Framework," http://www.osmf.org/.

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

[14] PPStream, "PPStream Homepage," http://www.pps.tv.

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

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

[17] Mea Wang and Baochun Li, "R2: Random push with random network coding in live peer-to-peer streaming," *IEEE J. Select. Areas Commun.*, vol. 25, pp. 1655–1666, Dec. 2007.

[18] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, December 2007.

[19] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in *Proceedings of ACM SIGCOMM*, 2008.

[20] J. C. Yang-hua Chu and H. Zhang.

[21] N. S. Dave Levin, Katrina Lacurts and B. Bhattacharjee.

[22] BitTorrent, "Homepage," http://www.bittorrent.com.

[23] M. J. F. Christina Aperjis and R. Johari, "Peer-assisted content distribution with prices," in *Proceedings of ACM CoNEXT*, 2008.

[24] C. Buragohain, D. Agrawal, and S. Suri, "A game theoretic framework for incentives in p2p systems," in *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, 2003.

[25] V. Misra, S. Ioannidis, A. Chaintreau, and L. Massoulié, "Incentivizing peer-assisted services: a fluid shapley value approach," in *Proceedings of ACM SIGMETRICS*, 2010.

[26] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

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

[28] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proceedings of ACM SIGCOMM*, 2004.

[29] Y. Zhou, D.-M. Chiu, and J. C. S. Lui, "A simple model for chunk-scheduling strategies in p2p streaming," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 42–54, 2011.

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

[31] G. Tian and Y. Liu, "Towards agile and smooth video adaption in dynamic http streaming," in *Proceedings of ACM CoNEXT*, 2012.