# Incentive Cooperation Strategies for Peer-to-Peer Live Multimedia Streaming Social Networks

W. Sabrina Lin, *Member, IEEE*, H. Vicky Zhao, *Member, IEEE*, and K. J. Ray Liu, *Fellow, IEEE*

***Abstract*—Multimedia social networks have become an emerging research area, in which analysis and modeling of the behavior of users who share multimedia are of ample importance in understanding the impact of human dynamics on multimedia systems. In peer-to-peer live-streaming social networks, users cooperate with each other to provide a distributed, highly scalable and robust platform for live streaming applications. However, every user wishes to use as much bandwidth as possible to receive a high-quality video, while full cooperation cannot be guaranteed. This paper proposes a game-theoretic framework to model user behavior and designs incentive-based strategies to stimulate user cooperation in peer-to-peer live streaming. We first analyze the Nash equilibrium and the Pareto optimality of two-person game and then extend to multiuser case. We also take into consideration selfish users' cheating behavior and malicious users' attacking behavior. Both our analytical and simulation results show that the proposed strategies can effectively stimulate user cooperation, achieve cheat free, attack resistance and help to provide reliable services.**

***Index Terms*—Cooperation strategy, game theory, multimedia live streaming, peer-to-peer, social network.**

## I. Introduction

With recent advance in networking, multimedia signal processing, and communication technologies, we witness the emergence of large-scale multimedia social networks, where millions of users form a distributed and dynamically changing infrastructure to share media content. Peer-to-peer (P2P) live streaming using the mesh-pull architecture [1] is one of the biggest multimedia social networks on the Internet and has enjoyed many successful deployments to date, for example, CoolStreaming, pplive and SopCast [2]–[5]. Users in a P2P live-streaming system watch live broadcasting TV programs over networks simultaneously. The system relies on voluntary contributions of resources from individual users to achieve high scalability and robustness and to provide satisfactory performance. Cooperation also enables users to access extra resources from their peers and thus benefit each individual user as well.

Manuscript received May 30, 2008; revised November 25, 2008. Current version published March 18, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lexing Xie.

W. S. Lin and K. J. R. Liu are with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: wylin@umd.edu; kjrliu@umd.edu).

H. V. Zhao is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: vzhao@ece.ualberta.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2009.2012915

However, due to the voluntary participation nature and the limited resources, full user cooperation cannot be guaranteed. A recent study showed that many users in Napster and Gnutella are free riders and 25% of them share no files at all. Therefore, an essential issue to be resolved first is to stimulate user cooperation. In addition, users in P2P live streaming systems are strategic and rational, in that they are likely to manipulate any incentive system (for example, by cheating) to maximize their payoff. As such, in large-scale social networks, users influence each other's decisions and performance, and there exist complicated dynamics among users. It is of ample importance to investigate user behavior and analyze the impact of human factors on multimedia social networks.

In the literature, there have been a lot of work on providing incentives for cooperation in P2P file sharing. BitTorrent [6], one of the most popular peer-to-peer file downloading system, adopts the tit-for-tat strategy to motivate user cooperation and prevent free riding [7]–[9]. A micro-payment mechanism was proposed in [10], where users earn rewards by uploading to others and pay for downloading. Using a game-theoretic model, the work in [10] showed that there exists equilibrium for micro-payment mechanisms. The work in [11] used game theory to model the interactions of peers and proposed a differential service-based incentive scheme, and in [12], the generalized prison's dilemma was used to model the P2P system.

However, designing incentive mechanisms for live video streaming is more challenging. Different from prior work on P2P file-sharing, P2P live streaming has many unique issues that need to be addressed. First, users receive payoff not only from the availability of files, but also from the ability to obtain high-quality streams. In addition, the delay constraint is much more stringent in P2P streaming, and a packet must arrive before its playtime to contribute to the increment of user's payoff.

To address the unique challenges in providing incentives in P2P live streaming, a rank-based peer-selection mechanism was introduced in [13], and in [14], a payment-based incentive mechanism was proposed, where peers pay points to receive data and earn points by forwarding data to others. Both work above use reputation or micro-payment based mechanisms, which often demand a centralized architecture thus hinder their scalability. The work in [15] proposed a distributed incentive mechanisms on mesh-pull P2P live video streaming systems.

The above prior work on incentive mechanisms for P2P live streaming either relied on trusted central billing services to implement micro-payment, or they assumed that all users are rational and honest. In real-world social networks, there are al-


Fig. 1. User dynamics in real-world social networks.

ways users with different objectives and everyone wants to maximize his or her own payoff as in Fig. 1. Some users will try by all means to achieve maximum utility, and they will *cheat* other users if they believe cheating can help improve their payoffs. In addition, there might also exist *malicious* users who aim to exhaust others' resources and attack the system. For example, they can tamper the media files with the intention of making the content useless (the so-called "pollution" attack) [16]. They can also launch the denial of service (DoS) attack to exhaust other users' resources and make the system unavailable [17]. Therefore, cheat prevention and attack resistance are fundamental requirements in order to achieve user cooperation and provide reliable services.

In this paper, we will focus on designing distributed, cheat-proof and attack-resistant cooperation stimulation strategies for P2P live streaming social networks under a game theoretic framework. We first consider a simple scenario with nonscalable video coding and study a game with only two players. We investigate the Nash equilibriums of the game and derive cheat-proof stimulation strategies. This analysis aims to stimulate each pair of users in P2P live streaming to cooperate with each other and achieve better performance. Then, we address the issue of cooperation stimulation among multiple users with nonscalable video coding, and investigate cheat-proof and attack-resistant incentive mechanisms. Finally, we design a chunk-request algorithm to maximize users' video quality when the layered video coding is used, which is the unique issue in P2P live streaming. We combine the algorithm together with our proposed cheat-proof and attack-resistant strategies to provide incentives for cooperation. Our proposed cheat-proof and attack-resistant mechanism rewards users who contribute more with more video chunks and thus better quality. It includes a request-answering algorithm for the data supplier to upload more to the peers from which it downloads more, and a chunk-request algorithm for the requestor to address the tradeoffs among different quality measure and to optimize the reconstructed video quality.

There are several major differences to distinguish our work from existing works listed above. First, we model the P2P live streaming as a multimedia social network, use game theory to model the dynamics among social network members, and propose cooperation stimulation strategies based on the analysis of behavior dynamics. Secondly, we address these issues under more realistic scenarios where the quality of the Internet connections among different users may vary, and provide a fully distributed solution while most existing cooperation schemes enforce every user to cooperate through central authorities. Thirdly, we fully explore potential cheating and malicious behavior, and derive cheat-proof and attack-resistant strategies while most existing work require nodes to honestly report their private information.

The rest of this paper is organized as follows. Section II introduces the mesh-pull P2P live streaming system model, studies the two-player game model, and analyzes the Nash equilibriums. In Section III, we propose a cheat-proof and attack-resistant strategy to stimulate user cooperation among all peers in P2P live streaming, and prove that it achieves Nash equilibrium, Pareto optimality, and subgame perfectness. Section IV proposes a cheat-proof and attack-resistant incentive mechanism with layered video coding. Section V shows simulation results to evaluate the performance of the proposed strategies. Finally, Section VI concludes this paper.

## II. Optimal Strategies in a Two-Player P2P Live Streaming Game

In this section, we first describe how two users in a P2P live streaming social network cooperate with each other. We then define the payoff function and introduce the game-theoretic formulation of user dynamics.

### A. Mesh-Pull P2P Live Streaming

We first introduce the basic protocol and streaming mechanisms of mesh-pull P2P live streaming system as in Fig. 2(a). In a mesh-pull delivery architecture for live video streaming [18], a compressed video bit stream of bit rate $B$ bps is divided into media chunks of $M$ bits per chunk, and all the chunks are available at the original streaming server. When a peer wants to view the video, he/she obtains a list of peers that are currently watching the video, and establishes partnership with several peers. At any instance, a peer buffers up to a few minutes worth of chunk within a sliding window. Each user keeps a "buffer map," indicating the chunks that he/she has currently buffered and can share with others, and they exchange their buffer maps with each other frequently. For example, in Fig. 2(b), peer 1 has first two chunks, while peer 2 has last two chunks, indicated by grey blocks in their video buffer maps. After peer 1 receives peer 2's buffer map, peer 1 can request one or more chunks that peer 2 has advertised in his/her buffer map. Time is divided into rounds of $\tau$ seconds. Fig. 2(b) shows how the peers cooperate with each other: at the beginning of each round, every user sends a chunk request either to one of his peers or to

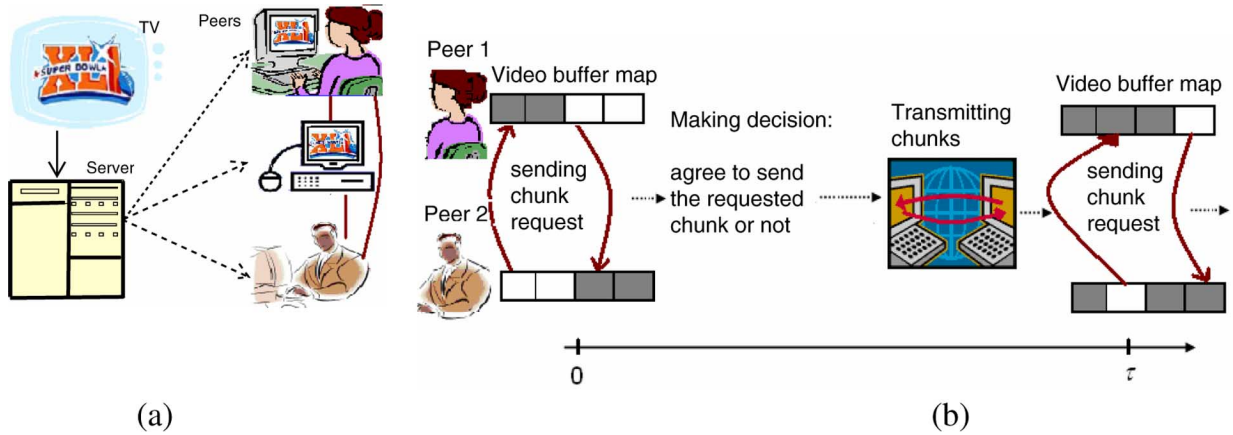


Fig. 2. Mesh-pull P2P Live Streaming Model. (a) Overall system. (b) Peer cooperation.

the original streaming server. Then, the supplier either replies with the requested chunk or rejects the request.

### B. Two-Player Game Model

We assume that there are totally $N$ users in the live streaming social network and every user buffers $L$ chunks. The video stream is originally stored in the streaming server whose upload bandwidth can only afford transmitting $K'$ chunks in one round ($\tau$ seconds) with $K' \ll N$. The server has no information of users' network topology, and the peer-to-peer system is information-pull, which means that the server only sends chunks that are requested by some users, and it replies the chunk requests in a round robin fashion. Due to the playback time lags among peers [18], different users request different chunks, and the server cannot answer all users' requests. In such a scenario, peers have to help each other to receive more chunks and thus better-quality videos.

This section investigates the incentive mechanisms for peer cooperation in live streaming. We start with a simple scenario with two cooperating users and nonscalable video coding structure. To simplify the analysis, in this paper, we consider a simple scenario where in each round, every peer can only request one chunk from the other peer and also uploads at most one chunk to the other.

We first define the utility (payoff) function of the two-player game. In each round, if player $i$ answers the other player $k$'s request and sends the requested data chunk to $k$, we define $i$'s cost $c_i$ as the percentage of his/her upload bandwidth used to transmit the requested chunk. That is, $c_i = M/(W_i\tau)$, where $W_i$ is player $i$'s total available upload bandwidth, $M$ is the size of the chunk and $\tau$ is the time duration of the round. If player $k$ forwards the data that $i$ requested and player $i$ receives the chunk correctly, then $i$ receives a gain of $g_i$, which is a user-defined value between 0 and 1. Every user in the P2P live streaming social network defines his/her own value of $g_i$ depending on how much he/she wants to watch the video. For instance, if all the user does is watching the live streaming and not distracted by other activities, $g_i$ can be chosen as 1, which also implies that the user is willing to cooperate with others to get the better-quality video. On the other hand, if the user is watching several videos, browsing the Internet, or downloading files simultaneously, he/she will not value the live streaming much thus set lower value $g_i$. Intuitively, if the user does not care about the video quality, $g_i$ would be set to 0 and the user will download the video directly from the server and not join the live streaming social network, since by joining the live streaming social network, some of his/her upload bandwidths would be occupied. We assume that $c_i$ is upper bounded by $c_{\max}$, which is the same as if there exists a minimum upload bandwidth $W_{\min}$ for all users such that $W_i \geq W_{\min}$. The minimum upload bandwidth constraint is necessary since if the user cannot even completely upload a chunk in one round period, other users have no incentive to cooperate with him/her. Here, $W_i$ and $g_i$ are player $i$'s private information, and it is not known to the other player unless player $i$ reports them.

Let the action of player $i$ takes *at each round* be $a_i$. In each round, player $i$ can choose its action $a_i$ from $0, 1$, where $a_i = 0$ means in this round, player $i$ chooses not to respond to the other player's request, while $a_i = 1$ indicates that player $i$ is willing to cooperate at this round. Let $P_{12}$ denote the probability that the chunk is successfully transmitted from user 1 to user 2, and $P_{21}$ is defined as the probability that user 2 successfully transmits the requested chunk to user 1. Then, for each round, provided that the action profile $(a_1, a_2)$ being taken, player 1 and 2's payoffs are calculated as follows:

$$\begin{aligned}\pi_1(a_1,a_2) &= (a_2P_{21})g_1 - a_1c_1 \\ &= (a_2P_{21})g_1 - a_1\frac{M}{W_1\tau} \\ \pi_2(a_1,a_2) &= (a_1P_{12})g_2 - a_2c_2 \\ &= (a_1P_{12})g_2 - a_2\frac{M}{W_2\tau}. \end{aligned} \quad (1)$$

The above payoff function consists of two terms: the first term in $\pi_i$ denotes the gain of user $i$ with respect to the other's action, and the second term denotes his/her cost with respect to his/her own action. From (1), it is reasonable to assume that $P_{21}g_1 \geq c_1$ and $P_{12}g_2 \geq c_2$, since users will only cooperate with each other if cooperation can benefit both users and give them positive payoffs. Let $\pi(a_1, a_2) = (\pi_1(a_1, a_2), \pi_2(a_1, a_2))$ be the payoff profile.

It is easy to check that, if this game will only be played for onetime, the only Nash equilibrium (NE) is (0,0), which means no one will answer the other's request. According to the backward induction principle [20], this is also true when the repeated game will be played for finite times with game termination time

known to both players. Therefore, in such scenarios, for each player, its only optimal strategy is to always play noncooperatively. However, in live streaming systems, these two players will interact many rounds and no one can know exactly when will the other user quit the game. Next, we show that cooperative strategies can be obtained under a more realistic setting. Let $\mathbf{s}_i = (a_i^{(1)}, a_i^{(2)}, \cdots)$ denote player $i$'s behavior strategy in the infinitely repeated game, where $a_i^{(j)}$ be the action that player $i$ takes at the $j$th round, and $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$ is the strategy profile. When the game is played more than one time, sum of payoff in every time should be considered as each players utility. However, in infinite time game model, sum of payoff usually goes to infinity, therefore, averaged payoff is considered instead. This means that we consider the following utility function of the infinitely repeated game:

$$U_i(\mathbf{s}) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \pi_i(\mathbf{s}). \tag{2}$$

Let us analyze the NEs for the infinitely repeated game with utility function $U_i$ defined as above. According to the Folk theorem [20], there exists at least one NE to achieve every feasible and enforceable payoff profile. A feasible payoff profile is the payoff that can be achieved; an enforceable payoff profile is the payoff that can be enforced by any mechanism to be achieved, which is, a feasible payoff profile that every players payoff is larger than or equal to zero. The set of feasible payoff profiles for the above game is

$$\begin{aligned} V_0 = \text{convex hull}\{(v_1, v_2) \,|\, &\exists\, (a_1, a_2) \text{ with} \\ &(\pi_1(a_1, a_2), \pi_2(a_1, a_2)) = (v_1, v_2)\}, \\ &\text{where } a_1, a_2 \in \{0, 1\} \end{aligned} \tag{3}$$

and the set of enforceable payoff, denoted by $V_1$

$$V_1 = \{(v_1, v_2) \,|\, (v_1, v_2) \in V_0 \quad \text{and} \quad v_1, v_2 \geq 0\}. \tag{4}$$

Fig. 3 illustrates the feasible and the enforceable regions of the above infinitely repeated game. The feasible region is inside the convex hull of $\{(0,0), (P_{21}g_1, -(M)/(W_2\tau)), (P_{21}g_1 - (M)/(W_1\tau), P_{12}g_2 - (M)/(W_2\tau)), (-(M)/(W_1\tau), P_{12}g_2)\}$. $V_1$ is the gray region shown in Fig. 3, which is the intersection of the feasible region and the first quadrant. It is clear that there exists an infinite number of Nash equilibriums. To simplify our equations, in this paper, we use $\mathbf{x} = (x_1, x_2)$ to denote the set of NE strategies corresponding to the enforceable payoff profile $(\pi_1(\mathbf{x}), \pi_2(\mathbf{x})) = (x_2 P_{21} g_1 - x_1(M)/(W_1\tau), x_1 P_{12} g_2 - x_2(M)/(W_2\tau))$. Intuitively, the NE strategy $x_i$ can be viewed as the averaged action that player $i$ takes over all rounds in the infinite game. Thus $x_i = \lim_{T\to\infty} \sum_{j=0}^{T} a_i^{(j)}/T$, and $0 \leq x_i \leq 1$.

### C. Nash Equilibrium Refinement

From the above analysis, one can see that the infinitely repeated game has infinite number of Nash equilibriums, and apparently, not all of them are simultaneously acceptable. For example, the payoff profile (0,0) is not acceptable from both players' point of view. Therefore, in this section, we will discuss how to refine the equilibriums based on new optimality criteria to eliminate those less rational Nash equilibriums and find out which equilibrium is cheat-proof. In this section, we consider the most widely used optimality criteria in the literature [21], [20]: Pareto optimality, proportional fairness, and absolute fairness.

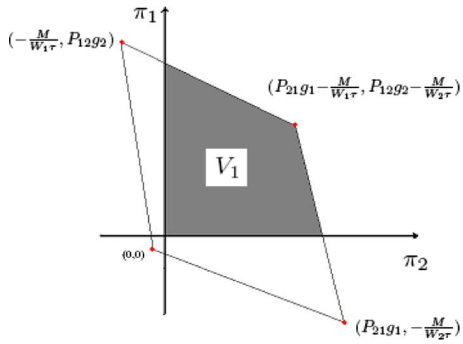


Fig. 3. Feasible and enforceable payoff profiles.

*1) Pareto Optimality:* A payoff profile $v \in V_0$ is Pareto optimal if and only if there is no $v' \in V_0$ such that $v_i' \geq v_i$ for all $i \in N$ [21]. Pareto optimality means no one can increase his/her payoff without degrade other's, which the rational players will always go to. It is clear from Fig. 3 that the line segment between $(P_{21}g_1, -(M)/(W_2\tau))$ and $(P_{21}g_1 - (M)/(W_1\tau), P_{12}g_2 - (M)/(W_2\tau))$ in the first quadrant and the line segment between $(-(M)/(W_1\tau), P_{12}g_2)$ and $(P_{21}g_1 - (M)/(W_1\tau), P_{12}g_2 - (M)/(W_2\tau))$ in the first quadrant is the Pareto-optimal set.

*2) Proportional Fairness:* Next, we will further refine the solution set based on the criterion of proportional fairness. Here, a payoff profile is proportionally fair if the product $U_1(s)U_2(s)$ can be maximized, which can be achieved by maximizing the product $\pi_1(x)\pi_2(x)$ in every round. It has been shown that the proportional fairness solution is always Pareto optimal. The proportional fairness point $x^* = (x_1^*, x_2^*)$ can be derived by solving

$$\begin{aligned} \max_{x_1, x_2} f(x_1, x_2) &= x_1 x_2 (P_{12} P_{21} g_1 g_2 + c_1 c_2) \\ &\quad - x_1^2 c_1 P_{12} g_2 - x_2^2 c_2 P_{21} g_1 \\ \text{s.t.} \qquad & 0 \leq x_1, x_2 \leq 1. \end{aligned} \tag{5}$$

In (5), same as in (1), $c_i = M/(W_i\tau)$ for $i = 1, 2$. It can be easily shown that the objective function $f(x_1, x_2)$ and the constraint functions are continuously differentiable at any feasible points, satisfying the Karush–Kuhn–Tucker conditions [23]. Thus the maximizer $(x_1^*, x_2^*)$ either satisfies $\nabla f(x_1^*, x_2^*) = 0$ or is on the boundary of the feasible region. If $\nabla f(x_1^*, x_2^*) = 0$, then $(x_1^*, x_2^*)$ satisfies

$$\begin{aligned} &\left.\frac{\partial \pi_1(x)\pi_2(x)}{\partial x_1}\right|_{(x_1^*, x_2^*)} \\ &\quad = x_2^* (P_{12} P_{21} g_1 g_2 + c_1 c_2) - 2 x_1^* P_{12} g_2 c_1 = 0 \\ &\left.\frac{\partial \pi_1(x)\pi_2(x)}{\partial x_2}\right|_{(x_1^*, x_2^*)} \\ &\quad = x_1^* (P_{12} P_{21} g_1 g_2 + c_1 c_2) - 2 x_2^* P_{21} g_1 c_2 = 0 \end{aligned} \tag{6}$$

which has only one solution $(x_1^* = 0, x_2^* = 0)$ with $f(0,0) = 0$. Apparently, it is not a desired solution. If $(x_1^*, x_2^*)$ is on the boundary of the feasible region, then it satisfies

$$\begin{aligned} x_1^* = 1,\ x_2^* &= \min\left\{1, \arg\max_{x_2} f(1, x_2)\right\} \\ &= \min\left\{1, \frac{P_{12}P_{21}g_1g_2 + c_1c_2}{2c_2P_{21}g_1}\right\} \\ \text{or}\quad x_2^* = 1,\ x_1^* &= \min\left\{1, \arg\max_{x_1} f(x_1, 1)\right\} \\ &= \min\left\{1, \frac{P_{12}P_{21}g_1g_2 + c_1c_2}{2c_1P_{12}g_2}\right\}. \end{aligned} \tag{7}$$

Combining (6) and (7), we can obtain the unique proportional fairness point in (8) at the bottom of the page.

*3) Absolute Fairness:* Although absolute fairness solution is not always Pareto-optimal, it is also an important criteria in many situations. Here we consider the absolute fairness in payoff, which refer to intuitively the most direct fairness criteria that the payoff of every player in the game is the same. By solving $\pi_1(x^*) = \pi_2(x^*)$, we can get the unique absolute fairness solution as in (9) at the bottom of the page.

### *D. Optimal and Cheat-Proof Strategies*

In Section II.C,we obtained several unique equilibriums with different optimality criteria. However, as in (8) and (9), all these solutions involve some private information $(g_i, W_i, P_{ji})$ reported by each player. Due to players' greediness, honestly reporting private information cannot be taken for granted and players may tend to cheat whenever they believe cheating can increase their payoffs.

*1) Cheat on Private Information $(g_i, W_i, P_{ji})$:* One way of cheating is to cheat on the private information $(g_i, W_i, P_{ji})$. First, let us examine whether the proportional fairness solution in (8) is cheat-proof with respect to $(g_i, W_i, P_{ij})$.

From (8), when

$$\frac{P_{12}P_{21}g_1g_2 + M^2/(W_1W_2\tau^2)}{2P_{21}g_1M/(W_2\tau)} = \frac{P_{12}g_2}{2M/(W_2\tau)} + \frac{M/(W_1\tau)}{2P_{21}g_1} \le 1 \tag{10}$$

$x_1^* = 1$ is fixed and

$$x_2^* = \frac{P_{12}g_2}{2M/(W_2\tau)} + \frac{M/(W_1\tau)}{2P_{21}g_1}. \tag{11}$$

From (11), if user 2 reports false and lower values of the product $P_{12}g_2W_2$, he/she can lower $x_2^*$ and, therefore, further increase his/her own payoff $\pi_2(1, x_2^*) = P_{12}g_2 - x_2^*(M)/(W_2\tau)$. Similarly, when

$$\frac{P_{12}P_{21}g_1g_2 + M^2/(W_1W_2\tau^2)}{2P_{12}g_2M/(W_1\tau)} = \frac{P_{21}g_1}{2M/(W_1\tau)} + \frac{M/(W_2\tau)}{2P_{12}g_2} \le 1 \tag{12}$$

$x_2^* = 1$ is fixed and

$$x_1^* = \frac{P_{21}g_1}{2M/(W_1\tau)} + \frac{M/(W_2\tau)}{2P_{12}g_2}. \tag{13}$$

By falsely reporting lower values of the product $P_{21}g_1W_1$, user 1 can lower $x_1^*$ and thus further increases his/her own payoff $\pi_1(x_1^*, 1) = P_{21}g_1 - x_1^*(M)/(W_1\tau)$. Therefore, the proportional fairness solution in (8) is not cheat-proof. Applying similar analysis on the absolute fairness solution in (9), we can also prove that the absolute fairness solution is also not cheat-proof with respect to private information. Therefore, players have no incentives to honestly report their private information. On the contrary, they will cheat whenever cheating can increase their payoff.

From the above analysis, to maximize their own payoffs, both players will report the minimum value of the product $P_{ji}g_iW_i$. Since we have assume that $P_{ji}g_i \ge c_i = M/(W_i\tau)$ and $W_i \ge W_{\min}$, both players will claim $P_{ji}g_iW_i = M/\tau$, and the solution (8) and (9) becomes

$$\mathbf{x}^* = (1, 1) \tag{14}$$

and the corresponding payoff profile is

$$\mathbf{v}^* = \left(P_{21}g_1 - \frac{M}{W_1\tau}, P_{12}g_2 - \frac{M}{W_2\tau}\right). \tag{15}$$

It implies that both players should always cooperate with each other. It is clear that the solution in (14) forms an Nash equilib-

$$x^* = \begin{cases} \left(\frac{P_{12}P_{21}g_1g_2 + M^2/(W_1W_2\tau^2)}{2P_{12}g_2M/(W_1\tau)}, 1\right), & \text{if } \frac{P_{12}P_{21}g_1g_2 + M^2/(W_1W_2\tau^2)}{2P_{12}g_2M/(W_1\tau)} \le 1 \\ (1,1), & \text{if } \frac{2}{P_{21}g_1W_1\tau/M + M/(P_{12}g_2W_2\tau)} \le 1 \le \frac{P_{12}g_2}{2M/(W_2\tau)} + \frac{M/(W_1\tau)}{2P_{21}g_1} \\ \left(1, \frac{P_{12}P_{21}g_1g_2 + M^2/(W_1W_2\tau^2)}{2P_{21}g_1M/(W_2\tau)}\right), & \text{if } \frac{P_{12}P_{21}g_1g_2 + M^2/(W_1W_2\tau^2)}{2P_{21}g_1M/(W_2\tau)} \le 1 \end{cases} \tag{8}$$

$$x^* = \begin{cases} \left(\frac{P_{21}g_1 + M/(W_2\tau)}{P_{12}g_2 + M/(W_1\tau)}, 1\right), & \text{if } P_{12}g_2 + \frac{M}{W_1\tau} \ge P_{21}g_1 + \frac{M}{W_2\tau} \\ \left(1, \frac{P_{12}g_2 + M/(W_1\tau)}{P_{21}g_1 + M/(W_2\tau)}\right), & \text{if } P_{21}g_1 + \frac{M}{W_2\tau} \ge P_{12}g_2 + \frac{M}{W_1\tau} \end{cases} \tag{9}$$

rium, is Pareto-optimal, and is cheat-proof with respect to private information $g_i$, $W_i$ and $P_{ji}$.

*2) Cheat on Buffer Map Information:* Here we assume every user has a buffer with fixed length $L$, which means the buffer stores $L$ future chunks. At the beginning of each round, each player has to exchange his/her own *buffer information* with the other player, that is, telling the other player which chunks he/she has in the buffer. The other way of cheating is to cheat on the buffer map information, that is, although player $i$ has the $k$th chunk, $\mathrm{LC}_k$, in the buffer, he/she tell the other player, player $j$, that he/she does not have $\mathrm{LC}_k$. By reporting this wrong buffer map information, the cheating user $i$ can reduce the number of requests from user $j$ since user $j$ will never ask for the cheated chunk $\mathrm{LC}_k$. As a result, increasing the cheating player's own payoff by lower $s_i$.

The only circumstance that cheating on buffer information is effective is that, when the cheated chunk $\mathrm{LC}_k$ is the only chunk that the honest player needs, and the honest user has other chunks that the cheater needs. Which means, the cheater can ask the honest user for help, but the honest user cannot ask the cheater for help because there is no chunk in the cheater's buffer that the honest user need. Under this circumstance, the cheater get the reward, but the honest user gets nothing. To prevent this kind of cheating, each player should not send chunks more than the other one sent.

To summarize, our *two-player cheat-proof P2P live streaming cooperation strategy* is as follows: in the two-player P2P live streaming game, in order to maximize each user's own payoff and be resistant to possible cheating behavior, a player should not send more chunks than its opponent does for it. Specifically, for each player in each round, it should always agree to send the requested chunk unless its opponent refused it in the previous round or there's no useful chunk in the opponent's buffer.

### *E. Performance of Two-Person Cheat-Proof Cooperation Strategy*

Here we study the performance of the two-player cheat-proof P2P live streaming cooperation strategy proposed above. In our simulation setting, there are totally 500 users in the network, and everyone is downloading chunks directly from the server. Each peer is either a DSL peer with 768 kbps uplink bandwidth, or a cable peer with 300 kbps uplink bandwidth. We fix the ratio between DSL peers and cable peers as 4:6. The video is initially stored at an original server with upload bandwidth of 3 Mbps. The request round is 1 second and each peer has a buffer that can store 30 s of video. We choose the "Foreman" video sequence with $352 \times 288$ spatial resolution at frame rate 30 frames per second and padding the video by another to two-hour long. A MPEG-4 video codec [24] is used to encode the video sequence into a nonscalable bit stream with bit rate 150 kbps. We divide the video into 1-s chunks, thus each chunk has $M = 150$ K bits. Among those peers, we randomly choose two who cooperate with each other using the proposed two-player cheat-proof P2P live streaming cooperation strategy. We set $g_1 = 1$, $g_2 = 1, 0.9, 0.8, 0.7$, and every peer claims the lowest bandwidth $W_{\min} = 300$ kbps.

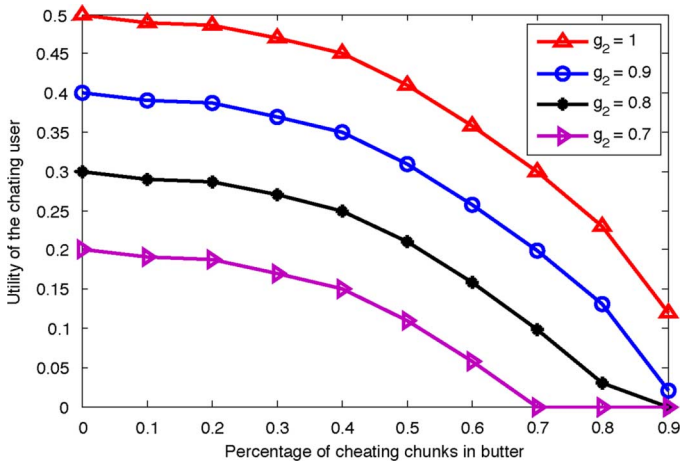


Fig. 4. Simulation results on two-person cheat-proof P2P live streaming cooperation strategy.

In our simulations, user 1 always reports accurate private information to user 2, and user 2 cheats on his/her buffer map information. Among all the chunks that user 2 received, he/she randomly selects $p_c$ percent of them, manipulates his/her buffer map, and tells user 1 that he/she does not have those selected chunks in the buffer. Fig. 4 shows the utility of user 2 with different gain $g_2$, where the x axis is $p_c$ and the y axis is the utility $v_i$. From Fig. 4, for a given $g_2$, a higher value of $P_c$ gives the cheating user a lower payoff. In addition, when g2 is small (for example, when $g_2 = 0.7$), if the cheating user selects a larger $p_c$, then he/she receives a zero payoff. That is, cheating cannot help a user increase his/her payoff, but rather reduces his/her utilities. It clearly demonstrates the cheat-proof property of our proposed strategy in Section II-D. In addition, from our simulations, by cooperating with each other, both peers double the number of chunks that they receive, which is 278 without cooperation and 542 after cooperation. Therefore, users can reconstruct a better-quality video.

## III. P2P Live Streaming Game

### *A. Multiuser Game Model*

Next, we will investigate how to stimulate cooperation for all members in peer-to-peer live streaming over heterogeneous and error-prone networks, and analyze users' behavior dynamics. We focus on the scenario that video streaming will keep alive for a relatively long time, and there exist a finite number of users, for example, people watch live Super Bowl over the Internet. Each user will stay in the social network for a reasonably long time, for example, from the beginning to the end of the game. They are allowed to leave and reconnect to the network when necessary. For each user, uploading chunks to other users will incur some cost, and successfully receiving chunks can improve the quality of his/her video and thus brings some gain. To simplify the analysis, in this section, we assume the video stream is encoded using nonscalable video codec. Therefore, for each user $i$, each received chunk gives the same gain $g_i$, whose value is specified by the user individually and independently. As discussed in Section II-B, $g_i$, the gain of receiving a chunk for the live video, is evaluated by user $i$ by how much he/she wants to

watch the video. For instance, $g_i$ should be set to 1 if at this moment, all user $i$ wants to do is watch the live streaming. The more activities user $i$ is doing simultaneously using the network bandwidth, the lower the $g_i$ is. If user $i$ is utilizing lots of his/her upload bandwidth and does not care about the quality of the live video stream, $g_i$ should be set to 0, and user $i$ will not join the P2P live stream social network.

In a real-world social network, some users may be malicious, whose goal is to cause damages to other users. In this paper, we focus on insider attackers, that is, the attackers also have legitimate identities, and their goal is to prevent the selfish users from getting chunks. In P2P live streaming social networks, there are two ways to attack the system.

1) **Incomplete chunk attack**: The malicious user agrees to send the entire requested chunk to the peer, but sends only portions of it or no data at all. By doing so, the requesting peer wastes his/her request quota in this round, and has to request the same chunk again in the next round.
2) **Pollution attack**: The other kind of attack in peer-to-peer live streaming is pollution [16]. In P2P streaming system, a malicious user corrupts the data chunks, renders the content unusable, and then makes this polluted content available for sharing with other peers. Unable to distinguish polluted chunks from unpolluted files, unsuspecting users download the polluted chunks into their own buffers, from which others may then download the polluted data. In this manner, polluted data chunks spread through the system.

Instead of forcing all users to act fully cooperatively, our goal is to stimulate cooperation among selfish users as much as possible and minimize the damages caused by malicious users. In general, not all cooperation decisions can be perfectly executed. For example, when a peer decides to send another peer the requested chunk, packets of the chunk may be dropped due to the overloaded routers. It is also possible that the chunk may fail to be completely received in one round due to the significant delay caused by the congested network. In this paper, we assume that the requesting peer gives up the chunk once it does not arrive in the round, and we use $p_{ij}$ to denote the probability of successful transmission of a chunk from peer i to peer j in one round of $\tau$ second. At the beginning of every round, each user will send only one chunk request to one user. Each user will respond to only one request. We assume every chunk request can be received immediately and perfectly.

In order to formally analyze cooperation and security in such peer-to-peer live streaming networks, we model the interactions among peers as the following game:

- **Server**: The video is originally stored at the original streaming server with upload bandwidth $W_s$, and the server will send chunks in a round-robin fashion to its peers.
- **Players and player type**: There are finite number of users/peers in the peer-to-peer live streaming social network, denoted by $N$. Each player $i \in N$ has a type $\theta_i \in \{\text{selfish}, \text{malicious}\}$. Let $N_s$ denote the set of all selfish players and $N_m = N \backslash N_s$ is the set including all insider attackers. A selfish user aims to maximize his/her own payoff, and may cheat other peers if cheating can help increase his/her payoff. A malicious user wishes to exhaust other peers' resources and attack the system.
- **Chunk requesting**: In each round, each player has *one* chunk-request quota, where he/she either *requests a chunk from a peer*, *requests a chunk from the video streaming source*, or *does not request any chunks* in this round.
- **Request answering**: For each player, after receiving a request asking for the upload of a chunk in its buffer, it can either *accept* or *refuse* the request.
- **Cost**: For any player $i \in N$, uploading a chunk to another player incurs cost $c_i = M/W_i\tau$, where $W_i$ is player $i$'s upload bandwidth and $W_i \geq W_{\min} \geq M/\tau$, same as in Section II-B.
- **Gain**: For each selfish user $i \in N_s$, if he/she requests a data chunk from another peer $j$, and if an unpolluted copy is successfully delivered to him/her, his/her gain is $g_i$, where $P_{ji}g_i > c_i$.
- **Utility function**: We first define the following symbols: for each player $i \in N$
  - $Cr^{(i)}(j,t)$ is the total number of chunks that $i$ has requested from $j$ by time $t$. Here, $j$ can be either a peer ($j \in N$) or $j$ is the streaming server. $Cr^{(i)}(t) = \sum_{j\in\{N,\text{ source}\}} Cr^{(i)}(j,t)$ denotes the total number of chunks that $i$ has requested by time $t$.
  - By time $t$, peer $i$ has successfully received $Cs^{(i)}(j,t)$ chunks from peer $j$ in time (a chunk is received in time if and only if it is received within the same round that it was requested). $Cs^{(i)}(t) = \sum_{j\in\{N,\text{source}\}} Cs^{(i)}(j,t)$ is peer $i$'s total number of successfully received chunks by time $t$.
  - By time $t$, $C_p^{(i)}(j,t)$ is the total number of polluted chunks that peer $i$ received from peer $j$. The total number of successively received unpolluted data chunks that peer $i$ received from peer $j$ is $Cs^{(i)}(j,t) - C_p^{(i)}(j,t)$, and each successfully received unpolluted chunk gives peer $j$ a gain of $g_i$.
  - $Cu^{(i)}(j,t)$ denotes the number of chunks that $i$ has uploaded to player $j$ by time $t$. $Cu^{(i)}(t) = \sum_{j\in\{N,\text{source}\}} Cu^{(i)}(t)$. The cost of uploading each chunk is $c_i$ for peer $i$.

Let $t_f$ be the lifetime of the peer-to-peer live streaming social network, and $T^{(i)}(t)$ denotes the total time that peer $i$ is in the network by time $t$. Then, we model the player's utility as follows.

1) For any selfish player $i \in N_s$, its utility $U_s^{(i)}(t_f)$ is defined as (16) at the bottom of the next page, where the numerator

$$U^{(i)}(t_f) = \frac{\left[Cs^{(i)}(t_f) - \sum_{j\in N} C_p^{(i)}(j,t_f)\right] g_i - Cu^{(i)}(t_f)\frac{M}{W_i\tau}}{Cr^{(i)}(t_f)} \tag{16}$$

denotes the net profit (i.e., the total gain minus the total cost) that the selfish peer $i$ obtained, and the denominator denotes the total number of chunks that $i$ has requested. This utility function represents the average net profit that $i$ can obtain per requested chunk, which $i$ aims to maximize.

2) For any malicious player $j \in N_m$, its objective is to maximize its utility in (17) at the bottom of the page. The numerator in (17) represents the net damage caused by j: the first term describes the total costs to other peers when sending the requested chunks to the malicious user $j$; the middle term evaluates other selfish peers' potential loss in gain due to the incomplete chunk attack by peer $j$; and the last term is peer $j$'s cost by uploading chunks to other peers. We normalize it using the lifetime of peer $j$, $T^{(j)}(t_f)$. Now, this utility function represents the average net damage that $j$ causes to the other nodes per time unit.

### *B. Cheat-Proof and Attack-Resistant Cooperation Stimulation Strategies*

Based on the system description in Section III-A, we can see that the multiple player game is much more complicated than the two-person game as in Section II, and pose new challenges. Thus, direct application of the two-player cooperation strategies to multiple player scenarios may not work.

*1) Challenges in Multiple User Scenario:* For peer-to-peer live streaming networks in heterogeneous Internet traffic environments, user cooperation stimulation has the following challenges.

- **Repeated game model is not applicable**. For example, a peer may request chunks from different peers at different times to maximize the utility. A direct consequence of such a nonrepeated model is that favors cannot be simultaneously granted. This makes cooperation stimulation in peer-to-peer live streaming networks an extremely challenging task.
- **Packet delay is inevitable** in Internet can cause severe trouble. For the two-player cheat-proof cooperation strategy, if the link between users is too busy that some packets of the chunk cannot arrive within a round time, the game will be terminated immediately and the performance will be degraded drastically. In addition, the malicious users can claim it was due to the erroneous Internet traffic and pretend to be nonmalicious. Distinguishing misbehavior caused by bit errors and packet loss from that caused by malicious intention is a challenging task.

*2) Credit Mechanism for Malicious User Detection:* To distinguish "intentional" malicious behavior from "innocent" misbehavior caused by packet delay, we introduce the credit mechanism. Addressing the pollution attack, for any two peers $i, j \in N$

$$Cc^{(i)}(j,t) = Cu^{(i)}(j,t) - C_p^{(j)}(i,t) \tag{18}$$

calculates the total number of *unpolluted* chunks that peer $i$ has uploaded to peer $j$ by time $t$. If the chunk is unpolluted, and is received before its playback time, then the chunk is useful. Note that for a selfish user $i \in N_s$, as discussed in the previous section, he/she has no incentives to intentionally send others polluted data chunks, since doing so will ultimately hurt himself/herself and lower the quality of his/her own video. However, since peer $i$ cannot identify a chunk as a polluted one until he/she starts decoding and playing that chunk, it is possible that user $i$ *unintentionally* forwards a polluted chunk to other peers. In this paper, addressing the above issue, we include the term $C_p^{(j)}(i,t)$ in (18) and consider the potential unintentional forwarding of polluted data chunks.

Given (18), we then define

$$\begin{aligned} D^{(i)}(j,t) &= Cc^{(i)}(j,t) - Cc^{(j)}(i,t) \\ &= \left(Cu^{(i)}(j,t) - C_p^{(j)}(i,t)\right) \\ &\quad - \left(Cu^{(j)}(i,t) - C_p^{(i)}(j,t)\right) \end{aligned} \tag{19}$$

which is the difference between the number of *useful* chunks that peer $i$ has sent to peer $j$ and the number of *useful* chunks that peer $j$ uploaded to peer $i$. Now, similar to the two-player cooperation-stimulation strategy in Section II-D, we consider the following strategy: each selfish peer $i \in N_s$ limits the number of chunks that he/she sends to any other peer $j$ such that by any time $t$, the total number of useful (unpolluted) chunks that $i$ has forwarded to $j$ should be no more than $Cu^{(j)}(i,t) - C_p^{(i)}(j,t) + D_{\max}^{(i)}(j,t)$, that is

$$D^{(i)}(j,t) \le D_{\max}^{(i)}(j,t), \quad \forall t \ge 0. \tag{20}$$

Here, $D_{\max}^{(i)}(j,t)$ is the "credit line" that user $i$ sets for user $j$ at time $t$. The credit line is set for two purposes: 1) to prevent egoism when favors cannot be simultaneously granted and to stimulate cooperation between $i$ and $j$, and 2) to limit the possible damages that $j$ can cause to $i$. By letting $D_{\max}^{(i)}(j,t) \ge 0$, $i$ agrees to send some extra, but at most $D_{\max}^{(i)}(j,t)$ chunks to $j$ without getting instant payback. Meanwhile, unlike acting fully cooperatively, the extra number of chunks that $i$ forwards to $j$ is bounded to limit the possible damages when $j$ plays noncooperatively or maliciously.

Player $i$'s goal of setting the credit line is to avoid helping player $j$ much more than player $j$ helps $i$ in long term's view, and vice versa, since neither of $i, j$ has incentive to send more chunks than the other does. Meanwhile, due to the dynamically changing network conditions, the request rates between $i$ and $j$ may vary from time to time. In this case, the credit line has to be large enough since a small credit line will refuse some requests even when the long-term average request rates between $i$ and $j$ are equal. The ultimate goal of setting the credit line is to make

$$U_m^{(j)} = \frac{\sum_{i \in N_s} Cu^{(i)}(j,t_f)\frac{M}{W_i \tau} + \sum_{i \in N_s}\left[Cr^{(i)}(j,t_f) - Cs^{(i)}(j,t_f)\right] P_{ji} g_i - Cu^{(j)}(t_f)\frac{M}{W_j \tau}}{T^{(j)}(t_f)} \tag{17}$$

sure that player $i$ and $j$ send asymptotically equal number of unpolluted chunks to each other, and

$$\lim_{t\to\infty} Cc^{(i)}(j,t) = \lim_{t\to\infty} Cc^{(i)}(j,t). \quad (21)$$

Combining the definition of $D^{(i)}_{\max}(j,t)$ with (21), $D^{(i)}_{\max}(j,t)$ must satisfy

$$\lim_{t\to\infty} \frac{D^{(i)}_{\max}(j,t)}{Cr^{(i)}(t)} = 0 \quad (22)$$

which also implies that arbitrarily increasing credit lines cannot always increase the number of accepted requests. Equation (22) provides an asymptotic upper bound for $D^{(i)}_{\max}(j,t)$. Based on the above analysis, to stimulate cooperation in the first few rounds, $D^{(i)}_{\max}(j,t)$ should be large enough in the first few cooperating rounds between user $i$ and $j$. On the other hand, $D^{(i)}_{\max}(j,t)/$[total number of rounds after time $t$] should be closed to 0 to prevent decreasing the utility of user $i$. Therefore, when choosing $D^{(i)}_{\max}(j,t)$, user $i$ should first estimate the number of remaining rounds for the live streaming, and choose a relatively small number $D_{\text{temp}}$. Then compare $D_{\text{temp}}$ with the reciprocal of $P_{ij}$, so that $D^{(i)}_{\max}(j,t)$ should be larger than $1/P_{ij}$ to stimulate the cooperation. A simple solution to this is to set the credit lines to be reasonably large positive constants, as in our simulations in Section V.

*3) Malicious User Detection:* Malicious attacks, such as the incomplete chunk attack and the pollution attack, exhaust other peers' resources and cause damages to the P2P live streaming system. Thus, it is of critical importance to implement a monitoring system to detect and identify misbehaving users, and a challenging issue is to differentiate "innocent" misbehavior (due to erroneous and congested networks) from "intentional" ones (for example, intentional pollution attacks).

If the credit line is set properly to satisfy (22), the damage of the pollution attack can be controlled to 0 asymptotically. Since the pollution attack will not effect honest users' utility by the credit line mechanism, in this section, we propose a malicious user detection algorithm that can differentiate the incomplete information attack to ensure the attack-resistance of the P2P live streaming social network.

$P_{ij}$ is the probability of successful transmitting one chunk within round period $\tau$. Hence when player $i$ decides to send a chunk to player $j$, with probability $1-P_{ij}$, this chunk transmission cannot be completed within one round because of packet dropping or delay caused by high traffic internet. That is, we use a Bernoulli random process to model the unsuccessful transmission of a chunk due to high traffic internet connection. Recall that $Cu^{(j)}(i,t)$ denote the number of chunks that $i$ has requested from $j$ and $j$ has agreed by time t, and $Cs^{(i)}(j,t)$ is the number of chunks that peer $i$ successfully receives from $j$ in one round. Given the Bernoulli random process, if user $j$ does not intentionally deploy the incomplete chunk attack, based on the Central Limit Theorem [25], for any positive real number $x$, we can have

$$\lim_{Cu^{(j)}(i,t)\to\infty} \text{Prob}\left(\frac{Cs^{(i)}(j,t) - Cu^{(j)}(i,t)P_{ji}}{\sqrt{Cu^{(j)}(i,t)Pji(1-P_{ji})}} \geq -x\right) = \Phi(x) \quad (23)$$

where $\Phi(x) = (1)/(\sqrt{2\pi})\int_{-\infty}^{x} e^{-t^2/2}dt$ is the Gauss tail function. If user $j$ does not intentionally sends incomplete chunks, (23) indicates that when the peer-to-peer live streaming game keeps going and $Cu^{(j)}(i,t)$ is large enough, then $Cs^{(i)}(j,t) - P_{ji}C^{(j)}_u(i,t)$ can be approximated by a Gaussian random variable with zero mean and variance $Cu^{(j)}(i,t)P_{ji}(1-P_{ji})$, that is

$$Cs^{(i)}(j,t) - Cu^{(j)}(i,t)P_{ji} \sim \mathcal{N}\left(0, Cu^{(j)}(i,t)P'_{ji}(1-P_{ji})\right). \quad (24)$$

Therefore, based on (24), given a predetermined threshold $h > 0$, every selfish peer $i$ can identify peer $j$ as a malicious user by thresholding $Cs^{(i)}(j,t) - Cu^{(j)}(i,t)P_{ji}$ as follows:

$$\begin{aligned} &j \in N^{(i)}_m(t) \quad \text{if and only if} \quad Cs^{(i)}(j,t) - Cu^{(j)}(i,t)P_{ji} \\ &\qquad \leq -h\sqrt{Cu^{(j)}(i,t)P_{ji}(1-P_{ji})}, \\ &\text{and} \quad j \in N^{(i)}_s(t) \quad \text{if and only if} \quad Cs^{(i)}(j,t) - Cu^{(j)}(i,t)P_{ji} \\ &\qquad > -h\sqrt{Cu^{(j)}(i,t)P_{ji}(1-P_{ji})}. \end{aligned} \quad (25)$$

In (25), $N^{(i)}_m(t)$ is the set of peers that are marked as malicious by peer $i$ at time $t$, and $N^{(i)}_s(t)$ is the set of peers that are marked as selfish by peer $i$ at time $t$. Based on (25), if the malicious user is always sending incomplete chunks to other users, then the probability of correctly identify the malicious user $(P_d)$ and the probability of falsely accusing a nonmalicious user as malicious $(P_{fa})$ can be written as

$$P_d = 1 - \Phi(h), \quad \text{and } P_{fa} = \Phi(h). \quad (26)$$

*4) Cooperation-Stimulation Strategies:* In reality, the interactions between peers are determined by the Internet topology and the communication pattern in the network. To analyze the effect of internet topology, we define $P_{ij}$ as the probability that peer $j$ successfully receives a chunk from peer $i$ in one round period $\tau$, $P_{si}$ denotes the probability of $i$ successfully receiving a chunk from the streaming server in one round period $\tau$, $P_s$ denotes the percentage of requests that the streaming server can answer in one round. These probabilities, $P_{ij}$, $P_{si}$, $P_s$, can be probed or estimated [15].

*Theorem 1:* For a selfish peer $i$, if

$$P_{si} \times P_s > P_{ji}, \quad \forall j \in N, j \neq i \quad (27)$$

then his/her optimal strategy is to always download the live video from the streaming server and to reject all chunk requests from other peers.

*Proof:* First, consider the optimal strategy for each round. At each round, peer $i$ has one chunk-request quota by which $i$ can ask a chunk either from the source or one peer $j \in N$. The probability that peer $i$ will successfully receive the requested chunk if $i$ sends request to source is $P_{si} \times P_s$, while the probability that peer $i$ will successfully receive the requested chunk if $i$ sends request to $j$ is $P_{ji}\times$ probability of j agrees to send the chunk. Obviously the probability of j agrees to send the chunk

is less than or equal to one, and since (27) holds, sending request to the source will give $i$ highest probability of getting the chunk/reward, which is the optimal chunk-request strategy in each round. Therefore, always asking chunks from the source is the optimal chunk-request strategy in the whole game. Also since peer $i$ always requests chunks from the original source, it does not have incentive to send any chunks to other peers in the network since it cost $M/W_i\tau$ to send a chunk which decreases peer $i$'s utility as in (16). From the above analysis, peer $i$ will always operate noncooperatively. □

Theorem 1 suggests that if a peer has a very good connection with the original streaming server, which is much better than the connections with all the other peers, then he/she will always refuse to cooperate. Cooperation cannot be enforced to these peers. But in real-world case, there are usually very few peers that can meet the above condition since peer-to-peer live streaming social networks are usually very big. Thus, the streaming server is often very busy with low $P_s$, and makes the condition $P_{si} \times P_s > P_{ji}$ for all $j \in N,\ j \neq i$ in Theorem 1 very difficult to satisfy.

The other extreme scenario is when peer $i$ is has the worst connection with other peers, that is, for every $j \in N,\ j \neq i$, there always exists another peer $k \in N,\ k \neq i, j$ such that $P_{ij} < P_{kj}$. In this scenario, will all the other peers in the network refuse to cooperate with him/her? The answer is no because of the dynamics in peer-to-peer social networks and the assumption of a busy server. Note that in peer-to-peer live streaming, different users have different playback time. If peer $i$'s playback time is earlier than all the other peers in the network, then it is very likely that his/her buffer has chunks that no other peers have, which is the incentive for other peers to cooperate with $i$ under the constraint that $D^{(j)}(i,t) \leq D^{(j)}_{\max}(i)$.

*5) Multiuser Attack-Resistant and Cheat-Proof Cooperation Strategy:* By summarizing the above results, we can arrive at the following cooperation stimulation strategies in peer-to-peer live streaming social networks.

**Multiuser attack-resistant and cheat-proof cooperation strategy**: in the peer-to-peer live streaming game, for any selfish peer $i \in N_s$ who does not meet the necessary condition (27) of Theorem 1, he/she initially marks every other user $j \in N,\ j \neq i$ as selfish. Then, in each round, $i$ uses the following strategy.

- If $i$ has been requested by $j$ to send a chunk, $i$ will accept this request if $j$ has not been marked as malicious by $i$ and (20) holds; otherwise, $i$ will reject the request.
- When $i$ is requesting a chunk, he/she will send the request to peer $j$ who satisfies

$$j = \arg \max_{j \in N_s^{(i)}(t), j \neq i} P'_{ji}. \tag{28}$$

- Let $1 - \Phi(h)$ be the maximum allowable false positive probability from $i$'s point of view, then, when $Cu^{(j)}(i,t)$ is large enough for any users $j \in N$, $i$ will apply the detection rule (25) to detect malicious behavior after each chunk request initiated by $i$.

### C. Strategy Analysis Under No Attacks

This section analyzes the optimality of the above proposed strategy for peers who do not satisfy the necessary conditions in Theorem 1 when there are no malicious users. We first consider an infinite-lifetime situation with $Cr^{(i)}(t) \to \infty$ as $t \to \infty$, and the finite-lifetime situation will be discussed later. First, we assume $D^{(i)}_{\max}(j,t)$ satisfies (22), which is to guarantee at most a finite number of $i$'s requests will be refused by $j$, and ensure $i$ needs $j$'s help the same as $i$ helps $j$ averagely.

*Lemma 1:* In the peer-to-peer live streaming game where some chunks may be dropped or delayed due to high traffic volume in the Internet, for a selfish player $j$, if all other users follow the multiuser attack-resistant and cheat-proofing cooperation strategy, then playing noncooperatively and sending only part of the requested chunks will not increase $j$'s payoff.

*Proof:* If user $j$ has agreed to upload a chunk to another user $i \in N$, by transmitting only part of the requested chunk will help $j$ reduce his/her cost. However, even though $j$ agrees to upload the chunk, it does not count as a successfully received chunk. In addition, player $i$ follows the multiuser attack-resistant and cheat-proof cooperation strategy, and always tries to let

$$\lim_{t\to\infty} Cs^{(i)}(j,t) \geq \lim_{t\to\infty} Cs^{(i)}(j,t). \tag{29}$$

Since (22) is satisfied, thus by sending partial of the requested chunk, player $j$ loses one chance to request a chunk from player $i$. To get this one-chunk-request chance back, player $j$ has to send another chunk completely and successfully to player $i$. Therefore, intentionally sending partial information of the requested chunks cannot bring any gain to player $j$. □

*Lemma 2:* For a selfish peer $i \in N_s$ in the peer-to-peer live streaming game with no malicious attackers, once $i$ has received a chunk request from another node $j \in N$, if (20) holds and if $j$ follows the multiuser attack-resistant and cheat-proofing cooperation strategy, then accepting the request is always an optimal decision from player $i$'s point of view.

*Proof:* From player $i$'s point of view, if (22) is satisfied, agreeing to send the requested chunk will not introduce any performance loss, since the average cost of helping $j$ goes to zero when $t \to \infty$. Meanwhile, refusing the request may cause $D^{(j)}(i,t) > D^{(j)}_{\max}(i,t)$ and thus forbids user $i$ to request chunks from player $j$ in the future. Therefore, accepting the request is an optimal decision. □

*Lemma 3:* In the peer-to-peer live streaming game with no malicious attackers, a selfish peer $i \in N_s$ has no incentive to cheat on his/her buffer map information.

*Proof:* From player $i$'s point of view, cheating on his/her buffer information will prevent other peers from requesting chunks from him/her, and thus will decrease the total number of chunks he/she needs to upload $(Cu^{(i)}(t))$. However, since other users always enforce (22) and $Cu^{(j)}(i,t) + D^{(j)}_{\max}(i,t) < Cu^{(i)}(j,t)$, decreasing $Cu^{(i)}(t)$ will also decrease the chance of getting chunks from other peers and lower player $i$'s overall payoff, similar to the two-player game in Section II-B. Therefore, selfish peers have no incentive to cheat on buffer information. □

*Theorem 2:* In the peer-to-peer live streaming game without malicious attackers, if all the selfish players who do not satisfy the necessary conditions in Theorem 1 follow the multiuser attack-resistant and cheat-proofing cooperation strategy forms a equilibrium with following properties: subgame perfect,

cheat-proof, and if $0 < \lim_{t\to\infty}(Cr^{(i)}(t))/(Cr^{(j)}(t)) < \infty$ for any $i, j \in N$, this equilibrium is also strongly Pareto optimal.

*Proof:*

1) **Cheat-proof**: Similar to the analysis of the two-person game in Section II, since no private information is involved in the game and Lemma 3 says that selfish users have no incentive to cheat on buffer information, we can conclude that the proposed cooperation-stimulation strategy is cheat-proofing.
2) **Nash equilibrium**: To prove that this strategy profile forms a subgame perfect equilibrium, note that this multiuser game can be decomposed into many two-player subgames. Therefore, we only need to consider the two-player subgame between player $i$ and $j$. Suppose that player $i$ does not follow the above strategy: either $i$ refuses to send chunks to player $j$ when (20) is satisfied; or $i$ intentionally sends only part of the chunk requested by player $j$; or $i$ sends more chunks than it should for player $j$, that is, $j$ agrees to send the requested chunks even (20) is not satisfied. First, from Lemma 1 and 2, neither refusing to sending chunks for other players when (20) is satisfied nor intentionally sending incomplete chunks will give player $i$ any performance gain. Secondly, sending many more chunks (i.e., more than $D^{(i)}_{\max}(j,t)$) than player $j$ has sent to $i$ will not increase player $i$'s own payoff either. This is because according to the assumption of credit line selections, $j$ will always cooperate with $i$ since $j$ has sent chunks less than $i$. Therefore, giving $j$ more favor will only cost $i$ more bandwidth. Based on the above analysis, we can conclude that the above multiuser attack-resistant and cheat-proofing cooperation strategy forms a Nash equilibrium.
3) **Subgame perfectness**: In every subgame of the equilibrium path, the strategies are: if player $j$ is marked malicious by peer $i$, player $j$ will play noncooperatively forever, which is a Nash equilibrium; otherwise, player $j$ follows the multiuser attack-resistant and cheat-proofing strategy, which is also a Nash equilibrium. Therefore, the proposed cooperation-stimulation strategy is subgame perfect.
4) **Strong Pareto optimality**: From the selfish user's utility function in (16), a player $i$ can either try to increase $Cs^{(i)}(t)$ or decrease $Cu^{(i)}(t)$ to increase his/her own payoff. However, from the above analysis, further decreasing of $Cu^{(i)}(t)$ will reduce other peers' successfully received useful chunks and therefore lower their payoff. In order to increase his/her payoff, the only thing that player $i$ can do is to increase $\lim_{t\to\infty} Cs^{(i)}(t)/Cr^{(i)}(t)$, which means that some other players will have to send more chunks to player $i$. Since all $Cr^{(i)}(t)$s are in the same order, increasing $\lim_{t\to\infty} Cs^{(i)}(t)/Cr^{(i)}(t)$ (and thus improving player $i$'s payoff) will definitely decrease the other players' payoff. Therefore, the above strategy profile is strongly Pareto optimal.

□

Until now, we have mainly focused on the situation that the game will be played for an infinite duration. In most situations, a peer will only stay in the network for a finite period of time, for example till the end of the video streaming. Then, for each player $i$, if $D^{(i)}_{\max}(j,t)$ is too large, he/she may have helped other users much more than his/her peers have helped $i$. Meanwhile, if $D^{(i)}_{\max}(j,t)$ is too small, he/she may lack enough peers to send chunks to him/her. How to select a good $D^{(i)}_{\max}(j,t)$ is a challenging issue. Section V will study the trade-off between the value of $D^{(i)}_{\max}(j,t)$ and the peers' utility through simulations. It is shows there that under given simulation scenarios, a relatively small $D^{(i)}_{\max}(j,t)$ value is good enough to achieve near-optimal performance, when compared to setting $D^{(i)}_{\max}(j,t)$ to be infinity. Here, it is also worth mentioning that the optimality of the proposed strategies cannot be guaranteed in finite-duration scenarios. However, we will show in the simulation results that the performance our cheat-proof and attack-resistant cooperation strategies is very closed to optimal.

### D. Strategy Analysis Under Malicious Attacks

In this section, we focus on the following two widely used attack models, the incomplete chunks attack and the pollution attack, and analyze the performance of the proposed cooperation-stimulation strategy when there exist malicious users. To simplify our analysis, we assume that $W_i = W$, $g_i = g$, and $g(M)/(W\tau) < \infty$ for all $i \in N$.

*Pollution Attack:* We first study the performance of the proposed strategy under the pollution attack. By always accepting selfish users' requests and sending polluted chunks to the selfish nodes, the malicious attackers can waste the selfish users' quota and prevent them from obtaining the gain of receiving useful chunks in that round. Note that every selfish user $i \in N_s$ forces $D^{(i)}(j,t) \le D^{(i)}_{\max}(j,t)$, calculates $D^{(i)}(j,t)$ as in (19), and does not include the polluted chunks in $D^{(i)}(j,t)$. Thus, for every selfish peer $i$, the damage cased by one pollution attacker is upper bounded by $D^{(i)}_{\max}(j,t)g$. Since $g < \infty$, as $t \to \infty$

$$\lim_{t\to\infty} \frac{D^{(i)}_{\max}(j,t)g}{Cr^{(i)}(t)} = 0 \tag{30}$$

and therefore, the overall damage due to pollution attacks becomes negligible.

*Incomplete Chunk Attack:* By sending incomplete chunks to others, malicious users inject trash traffic into the network and waste other peers' limited upload bandwidth. With the proposed attacker detection strategy in (25), for a malicious attacker to maximize the damages to the system, always sending incomplete chunks may not be a good strategy since it can be easily detected. Instead, to avoid being detected, attackers should selectively send incomplete chunks and send complete chunks in other time. According to the multiuser attack-resistant and cheat-proofing cooperation strategy in Section III-A, peer $j$ identifies $i$ as malicious if $Cs^{(j)}(i,t) - Cu^{(i)}(j,t)P_{ij} \le -h\sqrt{Cu^{(j)}(i,t)P_{ij}(1-P_{ij})}$. Assume that by time $t$, user $i$ has agreed to upload a total of $n$ chunks to user $j$. Therefore, to avoid being marked as malicious by $j$, $i$ has to successfully forward at least $nP_{ij} - h\sqrt{nP_{ij}(1-P_{ij})}$ complete chunks, and the maximum number of incomplete chunks that $i$ can send to $j$ is upper bounded by $n(1-P_{ij}) + h\sqrt{nP_{ij}(1-P_{ij})}$. Note that among these $n(1-P_{ij}) + h\sqrt{nP_{ij}(1-P_{ij})}$ incomplete chunks, $n(1-P_{ij})$ of them are dropped or delayed by the

network due the high Internet traffic volume, and the actual number of intentional incomplete chunks sent to $j$ by $i$ is bounded by $h\sqrt{nP_{ij}(1-P_{ij})}$. Therefore, for user $j$, the extra damaged caused by attacker $i$'s intentional malicious attack is upper bounded by $h\sqrt{nP_{ij}(1-P_{ij})}g$. Furthermore, to avoid being identified as malicious, attacker $i$ has to successfully forward at least $nP_{ij} - h\sqrt{nP_{ij}(1-P_{ij})}$ complete chunks to user $j$, which costs attacker $i$ a utility of $[nP_{ij} - h\sqrt{nP_{ij}(1-P_{ij})}](M)/(W\tau)$. Thus, following (17), the utility that attacker $i$ receives from intentionally sending incomplete chunks is at most $h\sqrt{nP_{ij}(1-P_{ij})}((M)/(W\tau)+g) - n(1-P_{ij})(M)/(W\tau)$. Since for any real positive $h$

$$\lim_{t\to\infty} \frac{h\sqrt{nP_{ij}(1-P_{ij})}\left(\frac{M}{W\tau}+g\right)}{n(1-P_{ij})\frac{M}{W\tau}} = 0 \tag{31}$$

selectively sending incomplete chunks can bring no gain to the attackers if they want to remain being undetected. In other words, if the game will be played for an infinite duration, sending incomplete chunks attack cannot cause damages to selfish nodes.

In summary, when the multiuser attack-resistant and cheat-proofing strategy is used by all selfish users, malicious attackers can only caused limited damage to the system. Further, the relative damage caused by the incomplete chunk attack will asymptotically approach zero when the game will be played for an infinite duration of time. Therefore, except some false alarm of identifying selfish users as malicious, selfish players' overall payoff will not be affected under attacks. From the above analysis, we can also see that no matter what objectives the attackers have and what attacking strategies that they use, as long as selfish peers apply the multiuser attack-resistant and cheat-proofing cooperation strategy, the selfish users' payoff and the overall system performance can be guaranteed.

*Optimal Attacking Strategy:* Based on the above analysis on the pollution attack and the incomplete chunk attack, we can conclude that, for the infinite-duration game, an attacker $j$'s overall payoff is upper bounded by

$$U_m^{(j)} \le \lim_{t\to\infty} \sum_{i\in N_s} \frac{D_{\max}^{(i)}(j,t)}{t} g \tag{32}$$

provided that all selfish users follow the multiuser attack-resistant and cheat-proof cooperation strategy. This upper bound can be achieved by the following *optimal attacking strategy in infinite game model*: in the peer-to-peer live streaming game, upon receiving a request an attacker $j \in Nm$ should always reject the requests; the attackers should always send requests to selfish users, until they do not agree to help.

When the game will only be played for a finite period of time, the above attacking strategy is not optimal any more. In addition to the pollution attack, the attackers can also send incomplete chunks without being detected. This is because the malicious attacker detection algorithm in Section III-B3 requires that the game has been played for a long time and peer $i$ and $j$ have interacted for a large number of times to provide an accurate estimation, and it will not be initiated unless $Cu^{(j)}(i,t)$ is large enough to avoid high false alarm rate. In such a scenario, different from the asymptotic analysis in (31), selfish users' performance will be degraded because of the incomplete chunk attack. However, in this paper we focus on the scenario the game will be played for a reasonably long time. Thus the users would have enough rounds to interact with each other and correctly estimate the statistics of chunk transmission, the relative damage is still insignificant.

## IV. P2P Live Streaming Game With Multiple Layered Coding

The previous section discussed the cheat-proofing and attack-resistant multiuser peer-to-peer live streaming cooperation-stimulation strategy with nonscalable video coding. In this section, we will extend the cooperation strategy to the scenario with layered video coding, where different chunks may belong to different layers and thus have different gain to the peers. In this scenario, an important issue is to schedule the chunk requests to maximize each peer's utility. We first investigate the chunk-request algorithm for a two-person P2P live-streaming social network that optimizes three different video quality measures in Section IV-B. We then propose a two-person chunk request algorithm considering tradeoff between these measures and extend it to N-person case. Then we will discuss the request-answering strategy when a peer $i$ receives more than one chunk requests at one round, and propose a cheat-proofing and attack-resistant cooperation strategy for P2P live streaming social networks.

### A. P2P Live Streaming With Scalable Video Coding

In P2P live streaming social networks, peers belong to different domains with different upload/download bandwidth, where scalable video coding is widely adopted to accommodate heterogenous networks [15], [26]. [15] shows that layered video coding provides higher quality of service in peer-to-peer live streaming social networks than multiple description coding (MDC) [27], thus we only consider the layered video coding. It decomposes the video sequence into different layers of different priority. The base layer contains the most important information of the video and is received by all users, and the enhancement layers gradually refine the reconstructed sequence at the decoder's side. Although scalable video coding provides service depending on peers' bandwidth capacity, it also has its unique challenges when used in P2P live streaming social network: the importance of different layers is unequal since higher layers cannot be decoded without successful decoding of lower layers. Therefore, in a P2P live streaming social network with scalable video coding, chunk-request algorithms need to assign higher priorities to the lower layers than to the higher layers.

In this paper, we encode a video into $L$ layers, and assume that the bit rate of every layer is the same $B_s$ bits/second. We further divide each layer into layer chunks (LCs) of $\tau$ seconds. Fig. 5 shows an example of the buffer map at one user's end. The grey blocks represent the chunks in buffer, while the white blocks denote the chunks that are not in the buffer, and "D" stands for layer chunks that are directly decodable after arriving. For

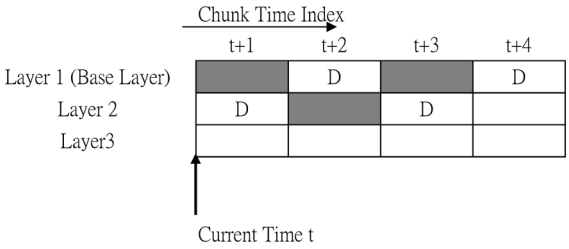


Fig. 5. Buffer map at a given time t.

example, this user only has the chunks in base layer with time index $\mathrm{t}+1$ and $\mathrm{t}+3$, and the chunk in layer 2 with time index $\mathrm{t}+2$ in buffer. A chunk is *decodable* if and only if all the lower layers in the same chunk time have been decoded correctly.

For user $j$, we define $a^{(j)}(t)$ as the number of decodable layer chunks at time $t$. For example, in the example in Fig. 5, $a^{(j)}(t+1)=1$, $a^{(j)}(t+2)=0$, $a^{(j)}(t+3)=1$, and $a^{(j)}(t+4)=0$. Let $T^{(j)}$ denote the duration that peer $j$ is in this network, then we define $A^{(j)}=\langle a_1^{(j)}, a_2^{(j)}, \cdots a_{T^{(j)}}^{(j)}\rangle$ as a vector containing all the $\{a^{(j)}(t)\}_{\{t=1,2,\ldots,T^{(j)}\}}$. $N^{(j)}$ is the number of all (decodable and successfully received nondecodable) chunks peer $j$ receives during his stay in the P2P live streaming social networks.

### B. Video Quality Measure

This paper focuses on investigating the best chunk-request strategy for each user in the peer-to-peer live streaming social network to optimize his/her own received video quality. In the literature, there are many video-quality measures. In this paper, we consider the following three popular criteria to evaluate our algorithms:

*Chunk Decodable Rate:* Every member in the P2P live-streaming social network has stringent bandwidth available, and every peer wants to use it as efficiently as possible. The chunk decodable rate $R^{(j)}$ of peer $j$ measures the bandwidth-efficiency of the chunk-request algorithm, and it is defined as

$$R^{(j)} \triangleq \frac{\sum_{i=1}^{T^{(j)}} a_i^{(j)}}{N^{(j)}}. \tag{33}$$

*Video Smoothness:* Intuitively, a video stream with nearly constant quality will be more pleasant to view than one with large swings in quality. Video smoothness measure (S) is defined as follows:

$$S^{(j)}(A) \triangleq \sum_{i=2}^{T^{(j)}} \left| \left( a_i^{(j)} - a_{i-1}^{(j)} \right) \right| \tag{34}$$

where $|\cdot|$ is the absolute value operator. $S^{(j)}(A)$ increases when the variance of $\{a^{(j)}(t)\}$ goes up, and decreases when the difference between adjacent $\{a^{(j)}(t)\}$ is lowered. To improve the quality and maximize the smoothness of the received video, user $j$ should request the chunks to minimize $S^{(j)}(A)$.

*Video Discontinuity Ratio:* Discontinuity ratio $\alpha^{(j)}$ of peer $j$ is defined as the percentage of times that a video is undecodable and unplayable. In a scalable video coding scheme, if all frames in the base layer are available, then the video is decodable and playable. Note that $a^{(j)}(t)$ stands for the number of chunks that is decodable at chunk time $i$. Therefore, if $a^{(j)}(t)=0$, peer $j$'s video is unplayable at chunk time $i$. $\alpha^{(j)}$ is defined as

$$\alpha^{(j)} \triangleq \frac{\sum_{i=1}^{T^{(j)}} \mathbf{U}(a_i^{(j)})}{T^{(j)}} \tag{35}$$

where $\mathbf{U}(a_i^{(j)})=1$ when $a_i^{(j)}>0$, otherwise $\mathbf{U}(a_i^{(j)})=0$.

### C. Optimal Chunk-Request Algorithms

In this subsection, we will propose three optimum chunk-request algorithms subject to the three video quality measures discussed in the previous section.

- **Maximizing Chunk-Decodable Rate:** We first discuss the chunk-request algorithm which aims to maximize the chunk decodable rate. According to the definition of chunk-decodable rate in (33), chunks that are not decodable do not give any gain to the player, thus gain of receiving the requested chunk $\mathrm{LC}(t',l)$ for player $j$ is

$$g_j = \begin{cases} g, & \text{if } \mathrm{LC}(t',l) \text{ is decodable} \\ 0, & \text{if } \mathrm{LC}(t',l) \text{ is not decodable} \end{cases} \tag{36}$$

  where $g>0$ is a constant, $t'$ is the time index of the requested layered chunk and $l$ stands for the layer index of the requested chunk. Therefore, maximizing payoff function in (1) is equivalent to making $g_i=g$, and it is to always requesting chunks that are directly decodable after arriving. In the example in Fig. 5, at the current state, requesting any one of the "D" chunks, $\mathrm{LC}(\mathrm{t}+1,\mathrm{layer2})$, $\mathrm{LC}(\mathrm{t}+2,\mathrm{layer1})$, $\mathrm{LC}(\mathrm{t}+3,\mathrm{layer2})$, and $\mathrm{LC}(\mathrm{t}+4,\mathrm{layer1})$, will maximize the player's payoff.
- **Maximizing Video Smoothness:** If the player concerns more about the video smoothness as defined in (34), the gain of receiving a requested chunk $\mathrm{LC}(t',l)$ for player $j$ is defined as the increment of smoothness after receiving the requested chunk as (37) at the bottom of the page, where $a_i'^{(j)}$ is the number of decodable layers in chunk time $i$ after receiving the requested chunk $\mathrm{LC}(t',l)$, and $t_0$ is the current playback time. The first term of the summation, $|a_i^{(j)}-a_{i-1}^{(j)}|$ represents the difference between the number of decodable layers in chunk time $i$ and then in $i-1$, hence $\sum_{i=t_0}^{i=t_0+L-1}|a_i^{(j)}-a_{i-1}^{(j)}|$ denotes the smoothness of the buffer map if the chunk LC(t',l) is not received. Similarly, $\sum_{i=t_0}^{i=t_0+L-1}|a_i'^{(j)}-a_{i-1}'^{(j)}|$ denotes the video smoothness if the requested chunk $\mathrm{LC}(t',l)$ is successfully received.

$$g_j = \begin{cases} \sum_{i=t_0}^{i=t_0+L-1} \left| a_i^{(j)} - a_{i-1}^{(j)} \right| - \left| a_i'^{(j)} - a_{i-1}'^{(j)} \right|, & \text{if } \mathrm{LC}(t',l) \text{ is decodable} \\ 0, & \text{if } \mathrm{LC}(t',l) \text{ is not decodable} \end{cases} \tag{37}$$

Therefore, to maximize the video smoothness, player $j$ should choose the decodable chunk that maximizes the difference $\sum_{i=t'}^{i=t'+1} |a_i^{(j)} - a_{i-1}^{(j)}| - |a_i'^{(j)} - a_{i-1}'^{(j)}|$ (with maxima greater than 0). If the maxima is less than 0, the peer should always choose undecodable chunks. Using the buffer map in Fig. 5 as an example, the peer should request $\mathrm{LC}(t+4, \text{layer } 1)$.

- **Minimizing Video Discontinuity Ratio:** If the peer wants to minimize video discontinuity ratio, the base layer is the most important and every chunk in base layer has equal importance according to the discontinuity definition in (35). Therefore, the gain of receiving a requested chunk $\mathrm{LC}(t', l)$ for player $j$ should be

$$g_j = \begin{cases} g, & \text{if } l = 1 \\ 0, & \text{if } l \neq 1. \end{cases} \tag{38}$$

To maximize $g_j$, the peer should request chunks in base layer. For the example in Fig. 5, requesting either $\mathrm{LC}(\mathrm{t}+2, \mathrm{layer1})$ or $\mathrm{LC}(\mathrm{t}+4, \mathrm{layer1})$ will maximize $g_j$.

The above three algorithms use different video quality measures defined in Section IV-B and select different chunks to maximize each individual criteria. To address the tradeoff between different video quality measure, we combine the above three chunk-request algorithms as follows.

*Step 1:* For user $j$, for each chunk $\mathrm{LC}(t', l)$ that is not in $j$'s buffer but is available at other peers' buffers, user $j$ assign a score $\mathrm{SC}(t', l)$ as follows.

- $j$ first assigns an original score $\mathrm{SC}(t', l) = ((t+L) - t')/L$ to the chunk $\mathrm{LC}(t', l)$, where $t$ is the current time and $L$ is user $j$'s buffer size. It addresses the stringent time constraint in video streaming, and gives the chunk $\mathrm{LC}(t', l)$ a higher score (thus higher priority for requesting) when it is closer to the playback time.
- If $\mathrm{LC}(t', l)$ is decodable after arriving, then the score is updated as $\mathrm{SC}(t', l) = \mathrm{SC}(t', l) + w_1$.
- If $g_2 = \sum_{i=t'}^{i=t'+1} |a_i^{(j)} - a_{i-1}^{(j)}| - |a_i'^{(j)} - a_{i-1}'^{(j)}| > 0$, then $j$ updates $\mathrm{SC}(t', l) = \mathrm{SC}(t', l) + w_2 g_2$.
- If $l = 1$, then $\mathrm{SC}(t', l) = \mathrm{SC}(t', l) + w_3$.

Here, $w_1 \geq 0, w_2 \geq 0, w_3 \geq 0$, $w_1 + w_2 + w_3 = 1$ are the weighs that the peer can adjust depending on the importance of each video-quality measure.

*Step 2:* Then, for each $\mathrm{LC}(t', l)$ that is not in $j$'s buffer but is available at other peers' buffers, let $\Omega(t', l)$ be the set of all users that who are not identified as malicious by user $j$, those who satisfy (20), and those who have $\mathrm{LC}(t', l)$ in their buffers. Then, user $j$ further updates the score of each chunk $\mathrm{LC}(t', l)$ as

$$\mathrm{SC}(t', l) = P_{kj}\mathrm{SC}(t', l), \quad \text{where} \quad P_{kj} = \max_{u \in \Omega(t', l)} P_{uj}. \tag{39}$$

*Step 3:* Finally, user $j$ selects the chunk with the highest score, that is, $(t^*, l^*) = \arg\max_{\{t', l\}} \mathrm{SC}(t', l)$, and requests the chunk $\mathrm{LC}(t^*, l^*)$ from peer $k$ who gives the highest successful transmission probability among all peers in $\Omega(t^*, l^*)$, that is, $k = \arg\max_u P_{uj}, u \in \Omega(t^*, l^*)$.

Since there is no algorithm bring optimal for all the three video quality measures, each peer can choose weights $w_1, w_2, w_3$ by themselves depending on which video-quality it concerns most.

### D. Request-Answering Algorithm

According to our analysis in Section III-B4, selfish users who do not satisfy the conditions of Theorem 1 should not reject chunk requests from other selfish peers, some peers may receive several chunk requests in a single round while our P2P environment assume that every user can upload at most one chunk per round. Thus we need a request-answering algorithm to address the above issue.

The peer-to-peer live streaming social network will last till the end of the video and has finite life time, selfish peers tend to consider the contributions from other peers when choosing which request to answer. This situation will encourage the selfish users to be always cooperative in the finite time model. Let $N_r^{(i)}(t) \subseteq N_s^{(i)}(t)$ be the set of users who send a chunk request to peer $i$ in round $t$ and all users in $N_r^{(i)}(t)$ are not marked as malicious by peer $i$, and also satisfy (22). We propose the following request-answering algorithm: for every selfish peer $i$, when he/she receives multiple chunk requests, he/she randomly chooses one peer $j$ with probability

$$P^{(i)}(j, t) = \frac{(Cs^{(i)}(j, t) + \epsilon)^{\gamma_i}}{\sum_{k \in N_r^{(i)}(t)} (Cs^{(i)}(k, t) + \epsilon)^{\gamma_i}} \tag{40}$$

where $\epsilon$ is a small number that gives newcomers who have not sent any chunks to peer $i$ a chance to start cooperation. $\gamma_i$ is a parameter that controls the sensitivity of peer $i$ to other peers' contribution. If $\gamma_i = 0$, every peer sent a request to peer $i$ has the same probability of being answered. On the contrary, if $\gamma_i \to \infty$, the request from peer who has send most chunks to peer $i$ will definitely be answered.

### E. P2P Live Streaming Cooperation Strategy With Layered Video Coding

From the above discussion, the **P2P live streaming cooperation strategy with layered video coding** is as follows: for any selfish node $i \in N_s$ who does not meet the necessary conditions of Theorem 1, initially $i$ marks every other nodes $j \in N$, $j \neq i$ as selfish. Then, in round $t$, $i$ uses the following strategy.

- In the chunk-requesting stage, $i$ chooses its own $(w_1, w_2, w_3)$, applies the chunk-request algorithm in Section IV-C, and sends one chunk request to one peer in $N_s^{(i)}(t)$.
- In the request-answering stage, $i$ first identifies the selfish peers that satisfies (22). Then, $i$ chooses a peer $j$ among them based on the probability distribution in (40), and agrees to send the requested chunk to $j$.
- Let $1 - \Phi(h)$ be the maximum allowable false positive probability from $i$'s point of view, then, as long as $Cu^{(j)}(i, t)$ is large enough for any node $j \in N$, $i$ applies the malicious user detection rule (25) after each chunk request that is initiated by $i$.

## V. Simulation Results

In our simulation, there are 200 DSL peer with 768 kbps uplink bandwidth and 300 cable peer with 300 kbps uplink band-

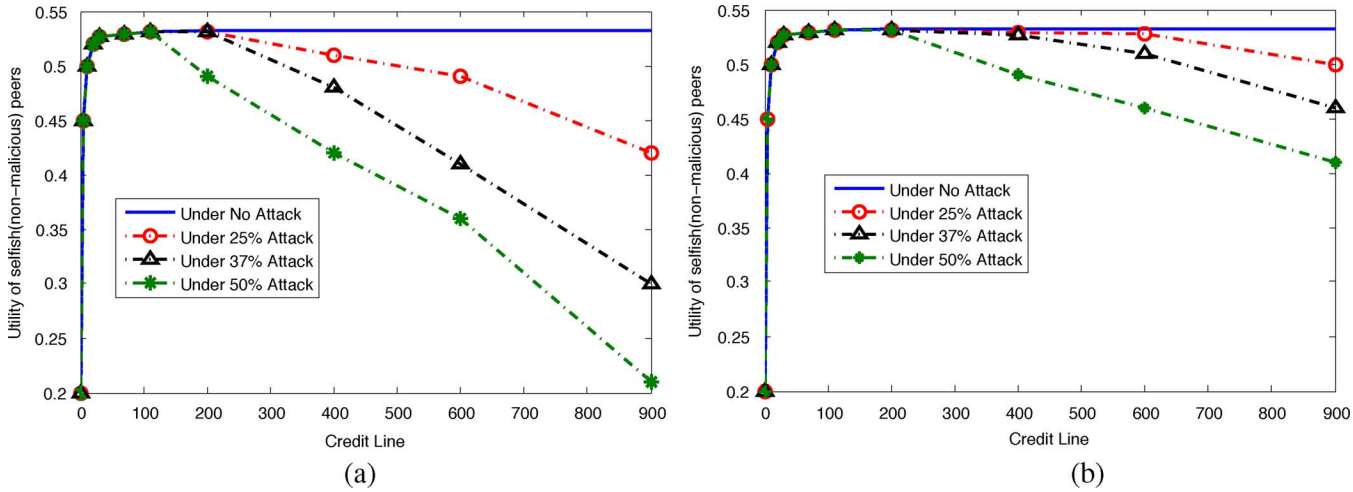


Fig. 6. Selfish peers' performance under proposed strategies with and without attack. (a) Three-layer video coding with $\tau = 0.4$ second. (b) Four-layer video coding with $\tau = 0.2$ second.

width. The video is initially stored at an original server with upload bandwidth 3 Mbps. We choose the "Foreman" video sequence ($352 \times 288$) resolution with frame rate 30 frame/sec and by attaching duplicated copies to the original video, make it into a 60 minutes video. Each user has a buffer with length 30 s. To exam the influence of different parameters on the performance of the proposed cooperation strategies, we run the simulations under two settings: First, we let the round duration $\tau$ is 0.4 s resulting in 9000 rounds in total for the P2P live streaming social network, and the video is coded into three-layer bitstream with 50 kbps per layer. Then the video is divided into 1-s layered chunks, thus chunk size $\mathrm{M} = 50$ kbits. In our second simulation setup, we let $\tau = 0.2$ second and the total number of rounds is $1.8 \times 10^4$. The video is encoded into four-layer bitstream with 37.5 kbps per layer. Each chunk is of 1-s length and includes M-37.5 K bits. We set the score weighing as $w_1 = 3/6, w_2 = 1/6, w_3 = 2/6$ and the malicious peers can either mount attack by sending incomplete or polluted chunks. The nonmalicious (selfish) peers follow the cheat-proof and attack-resistant cooperation strategies in Section IV-E.

We first study how different credit lines can affect cooperation stimulation. Fig. 6 demonstrates the relationship between the credit line when the percentage of attackers are 0, 25%, 37%, and 50%, respectively. The attackers are chosen randomly from all the 500 peers. Selfish peers follow the attack-resistant and cheat-proof cooperation strategy in Section IV-E, and the attackers follow the attack strategy in Section III-D. From these results, we can see that, in both simulation setups, when the credit line is over 50, the selfish nodes' payoffs are saturated. As the credit line keeps increasing, selfish nodes' utilities start to decrease very fast under attack. The selfish users' utilities remain the same if there are no attackers presented. It is clear from (32) that the maximum damage attackers can cause is linearly proportional to the credit line, while total number of rounds is 9000, when credit line is larger than 120 and 50% attackers, by (32), the damages are no longer negligible. Also, Fig. 6 suggests that setting credit line of 50 is an optimal choice for both simulation settings since it stimulates the cooperation to the maximum degree. Nevertheless, arbitrarily increasing credit line is dangerous for the selfish users since they do not know how many malicious users are in the network.

Next, we examine the robustness of our cooperation strategies against attackers and free-riders in terms of PSNR. Since from Fig. 6, both simulation settings give similar trends, here we use simulation setting 1 to demonstrate the robustness. Also, to show how the total number of users effects the optimal credit line, we test our proposed cooperation schemes on 500 users and 1000 users with fixed ratio between cable and DSL peers (3:2). We let the credit line equals to 50, 100, 200, or 300, respectively. Selfish peers follow the cooperation strategy in Section IV-E. Also the malicious peers are randomly selected and follow the optimal attack strategy in Section III-D. Fig. 7(a) and (b) shows the PSNR of a selfish user's video versus the percentage of attackers with different credit lines and different number of users. It is clear that when the credit line is chosen correctly, and is around 50, our cooperation strategies is attack-resistant in both cases. Even the credit line is too large, around 100, the PSNR of selfish users' video does not degrade too much even there are 60% malicious. From the above discussion, we can conclude that the optimal credit line is the value that just stimulates the cooperation, which should be around several dozen. If there are fewer users in the network, or the total number of rounds is larger, the range of attack-resistant credit line is larger. Although there is no explicit way to choose the credit line, in general, a credit line between 50 and 100 will simulate cooperation among selfish users, resist cheating behavior, and give good performance.

Fig. 7(c) shows the video quality (PSNR) of peers who follows our cooperation strategy with 500 users in Section IV-E and the free-riders versus percent of free riders. The credit line in Fig. 7(b) is 50. It is clear that there is no incentive for the peers to be free riders since their video quality is very bad, also our attack-resistant and cheat-proof cooperation strategy guarantees the peers' quality of service.

## VI. Conclusion

In this paper, we investigate cooperation stimulation in P2P live streaming social networks under a game theoretic

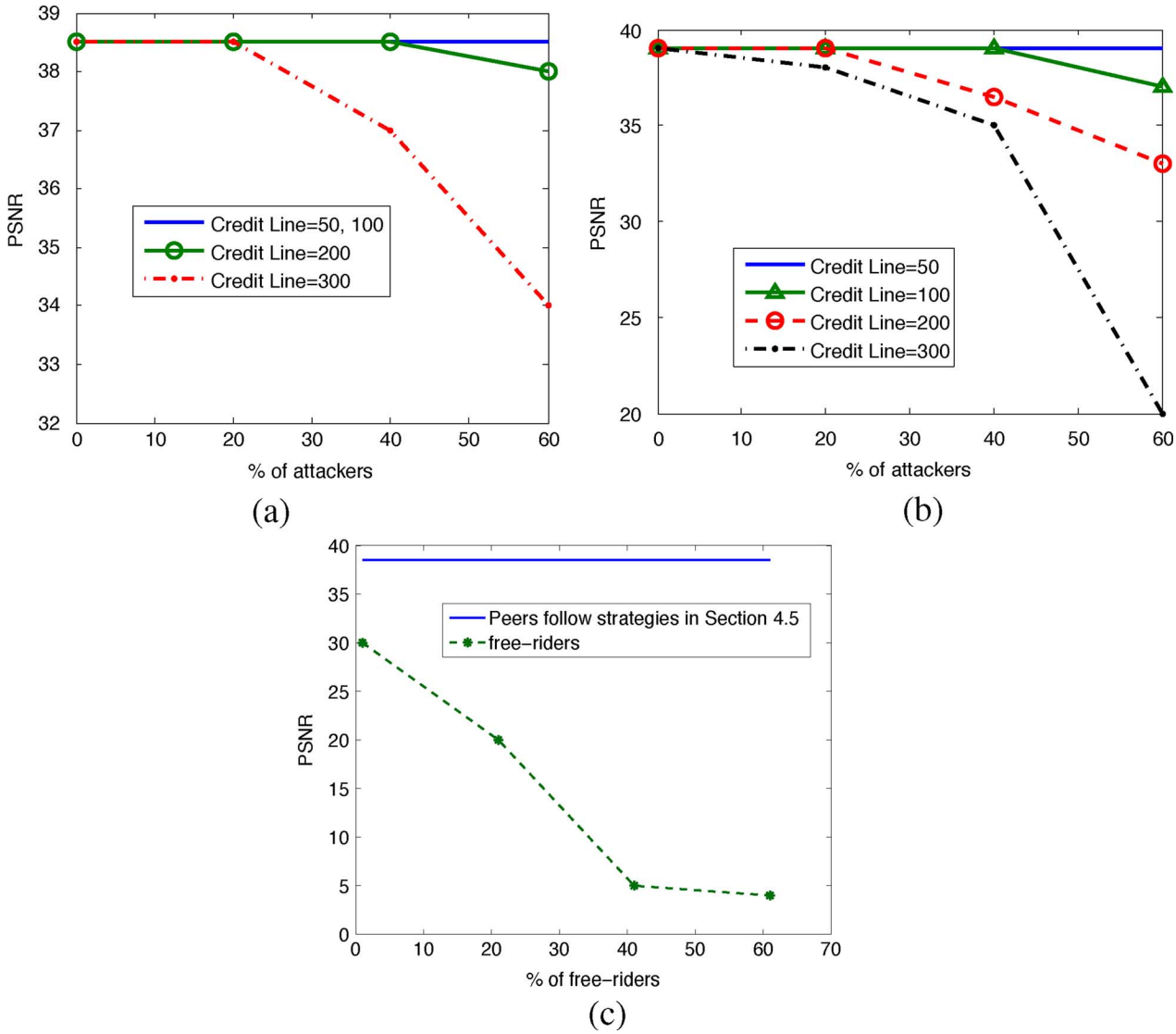


Fig. 7. Selfish peers' video quality (PSNR) versus the percentage of attackers and free-riders with (a) 500 users. (b) $10^3$ users. (c) Versus free-riders.

framework. Besides selfish behavior, possible attacks have also been studied, and attack-resistant cooperation stimulations have been devised which can work well under various traffic network and hostile environments. An illustrating two-player game is studied, and different optimality criteria, including Pareto-optimal, proportional fairness and absolute fairness is performed to refine the obtained Nash equilibriums. Finally, a unique Nash equilibrium solution is derived, which states that, in the two-person live streaming game, a node should not help its opponent more than its opponent has helped it.

The results are then extended to stimulate multiuser live streaming, and combing with the chunk-request and request-answering algorithm, a fully-distributed attack-resistant and cheat-proof cooperation stimulation strategy has been devised for P2P live streaming social networks. Simulation results have illustrated that the proposed strategies can effectively stimulate cooperation among selfish peers in internet with various traffic and hostile environments, and the chunk-request algorithm with tradeoffs performs the same as optimal algorithms when the percentage of attackers is lower than 20%.

## REFERENCES

[1] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.
[2] [Online]. Available: http://www.pplive.com/en/index.html.
[3] [Online]. Available: http://www.ppstream.com/.
[4] [Online]. Available: http://www.uusee.com/.
[5] X. Zhang, J. Liu, B. Li, and P. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," in *Proc. IEEE INFOCOM*, Mar. 2005, vol. 3, pp. 2102–2111.
[6] [Online]. Available: http://www.bittorrent.com/.
[7] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. 1st Workshop Economics of Peer-to-Peer Systems*, Jun. 2003.
[8] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. SIGCOMM*, 2004, pp. 367–378.
[9] S. Jun and M. Ahamad, "Incentives in BitTorrent induce free riding," in *Proc. 2005 ACM SIGCOMM Workshop Economics of Peer-to-Peer Systems*, 2005.
[10] P. Golle, K. Leyton-Brown, and I. Mironov, "Incentives for sharing in peer-to-peer networks," in *Proc. ACM Electronic Commerce (EC 01)*, Oct. 2001.
[11] C. Buragohain, D. Agrawal, and S. Suri, "A game theoretic framework for incentives in P2P systems," in *Proc. Int. Conf. Peer-to-Peer Computing*, Sep. 2003, pp. 48–56.

[12] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," in *Proc. ACM Conf. Electronic Commerce (EC-04)*, May 2004.
[13] A. Habib and J. Chuang, "Incentive mechanism for peer-to-peer media streaming," in *Proc. Int. Workshop Quality of Service (IWQoS)*, Jun. 2004, pp. 171–180.
[14] G. Tan and S. A. Jarvis, "A payment-based incentive and service differentiation mechanism for peer-to-peer streaming broadcast," in *Proc. Int. Workshop Quality of Service (IWQoS)*, Jun. 2006.
[15] Z. Liu, Y. Shen, S. Panwar, K. Ross, and Y. Wang, "Using layered video to provide incentives in P2P live streaming," in *Proc. ACM Special Interest Group on Data Communication*, Aug. 2007.
[16] J. Liang, R. Kumar, Y. Xi, and K. Ross, "Pollution in P2P file sharing systems," in *Proc. IEEE Conf. Computer Communications (INFOCOM)*, Dec. 2005, vol. 2.
[17] N. Naoumov and K. Ross, "Exploiting P2P systems for DDoS attacks," in *Proc. 1st Int. Conf. Scalable Information Systems*, 2006.
[18] X. Hei, Y. Liu, and K. W. Ross, "Inferring network-wide quality in P2P live streaming systems," *IEEE J. Select. Areas Commun.*, vol. 25, no. 9, pp. 1640–1654, Dec. 2007.
[19] W. Yu and K. J. R. Liu, "Game theoretic analysis of cooperation and security in autonomous ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 5, pp. 507–521, May 2007.
[20] M. J. Osborne and A. Rubinste*, A Course in Game Theory*. Cambridge, MA: The MIT Press, 1994.
[21] G. Owen*, Game Theory*, 3rd ed. New York: Academic, 1995.
[22] W. S. Lin, H. V. Zhao, and K. J. R. Liu, "A game theoretic framework for incentive-based peer-to-peer live-streaming social networks," in *Proc. IEEE Int. Conf. Acoustic, Speech, and Signal Processing (ICASSP)*, 2008.
[23] J. Nocedal and S. J. Wright*, Numerical Optimization*, 2nd ed. New York: Springer, 2000.
[24] A. Tourapis, K. SAuhring, and G. Sullivan, "Revised H.264/MPEG-4 AVC reference software manual," *Joint Video Team, Doc. JVT-Q042*, Oct. 2005.
[25] O. Kallenberg*, Foundations of Modern Probability*. New York: Springer-Verlag, 1977.
[26] D. Marpe, H. Schwarz, and T. Wiegand, "Joint scalable video model (JSVM) 2," *Joint Video Team, Doc. JVT-O202*, Apr. 2005.
[27] R. Puri and K. Ramchandran, "Multiple description source coding through forward error correction codes," in *Proc. 33rd Asilomar Conf. Signals, Systems and Computers*, Oct. 1999.

**W. Sabrina Lin** (M'06) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2002 and 2004, respectively. She is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of Maryland, College Park.

Her research interests are in the area of information security and forensics, multimedia signal processing, and multimedia social network analysis.

Ms. Lin received the University of Maryland Future Faculty Fellowship in 2007.

**H. Vicky Zhao** (M'05) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1997 and 1999, respectively, and the Ph.D. degree from the University of Maryland, College Park, in 2004, all in electrical engineering.

She was a Research Associate with the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park, from January 2005 to July 2006. Since August 2006, she has been an Assistant Professor with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. She coauthored the book *Multimedia Fingerprinting Forensics for Traitor Tracing* (New York: Hindawi, 2005). Her research interests include information security and forensics, multimedia, digital communications, and signal processing.

**K. J. Ray Liu** (F'03) is a Distinguished Scholar-Teacher of the University of Maryland, College Park. He is Associate Chair of Graduate Studies and Research of Electrical and Computer Engineering Department and leads the Maryland Signals and Information Group conducting research encompassing broad aspects of information technology, including communications and networking, information forensics and security, multimedia signal processing, and biomedical technology. His recent books include *Cooperative Communications and Networking* (Cambridge, U.K.: Cambridge University Press, 2008); *Resource Allocation for Wireless Networks: Basics, Techniques, and Applications* (Cambridge, U.K.: Cambridge University Press, 2008); *Ultra-Wideband Communication Systems: The Multiband OFDM Approach* (Piscataway, NJ: IEEE-Wiley, 2007); *Network-Aware Security for Group Communications* (New York: Springer, 2007); *Multimedia Fingerprinting Forensics for Traitor Tracing* (New York: Hindawi, 2005); and *Handbook on Array Processing and Sensor Networks* (Piscataway, NJ: IEEE-Wiley, 2009).

Dr. Liu is the recipient of numerous honors and awards, including best paper awards from IEEE Signal Processing Society (twice), IEEE Vehicular Technology Society, and EURASIP; IEEE Signal Processing Society Distinguished Lecturer, EURASIP Meritorious Service Award, and National Science Foundation Young Investigator Award. He also received various teaching and research recognitions from the University of Maryland, including university-level Invention of the Year Award; and Poole and Kent Senior Faculty Teaching Award and Outstanding Faculty Research Award, both from A. James Clark School of Engineering Faculty. He is a Fellow of AAAS. He is Vice President–Publications and on the Board of Governors of the IEEE Signal Processing Society. He was the Editor-in-Chief of IEEE SIGNAL PROCESSING MAGAZINE and the founding Editor-in-Chief of *EURASIP Journal on Applied Signal Processing*.