Foresighted Resource Reciprocation Strategy in BitTorrent Systems

UCLA Computer Science Department Technical Report 10-0014

Rafit Izhak-Ratzin
Computer Science Department
University of California
Los Angeles

Hyunggon Park
Electrical Engineering Institute
Ewha Womans University
Seoul, Korea

Mihaela van der Schaar
Electrical Engineering Department
University of California
Los Angeles

***Abstract*—Recent research efforts have shown that the popular Bit-Torrent protocol does not provide fair resource reciprocation and allows free-riding. In this paper, we propose a novel *foresighted resource reciprocation* mechanism that replaces the peer selection mechanism with a reinforcement learning mechanism that adopts a foresighted resource reciprocation policy. We model the peer interactions in the BitTorrent-like network as a *stochastic-game*, where we explicitly consider the strategic behavior peers. The peers can observe partial historic information of associated peers' statistical reciprocal behaviors, through which the peers can estimate the impact on their expected utility and then adopt their best response. The policy determines the peer's optimal resource reciprocations, and enables the peer to maximize the long-term performance. The mechanism improves fairness as it relies on long-term history. Moreover, it hurts the free-riders directly since foresighted peers are discouraged to upload to free-riders.**

We have implemented the proposed mechanism in an existing BitTorrent client. We have also performed extensive experiments on a controlled PlanetLab testbed to evaluate the mechanism effectiveness. Our results confirm that the foresighted resource reciprocation mechanism promotes fairness, improves the system robustness, and discourages free-riding in compare to the regular BitTorrent.

I. Introduction

In P2P content distribution systems, fairness among peers participating in content distribution is an important factor, as it encourages peers to actively collaborate in disseminating content, which can lead to improved system performance. However, even BitTorrent [2], one of the most popular protocols used in P2P content distribution, does not provide fair resource reciprocation, especially in node populations with heterogeneous upload bandwidths [3]–[6]. This is because the tit-for-tat strategy, that is implemented in BitTorrent, is based on short-term history, i.e., upload decisions are made based on most recent resource reciprocation observation. Moreover the decision is based on backward looking and not forward looking. Thus, a peer can follow the tit-for-tat policy only if it continuously upload pieces of a particular file and as long as it receives pieces of interest in return. However, this is not always possible, as peers can have no pieces in which the other peers are interested in, regardless of their willingness to cooperate [7]; yet, this behavior is still perceived as a lack of cooperation.

Additionally, it has been shown that BitTorrent systems do not effectively cope with selfish peer behaviors, such as free-riding [8]–[10], due to the optimistic unchoke mechanism currently used in BitTorrent. This enables peers to continuously discover better leechers to reciprocate resources with. However, it creates a major opportunity for peers to obtain data, without uploading in return. Moreover, it may induce unfairness in the system, as it forces high-capacity peers to interact with low-capacity ones.

Reputation-based schemes that are based on the propagation of global history (e.g., [11]–[13]) have been proposed to overcome the limitations of pure tit-for-tat and optimistic unchoke mechanisms. However, these approaches require significant communication overhead to maintain the global history across peers. Moreover, the reliability of global history is unclear as peers may exhibit different reciprocation behaviors with different peers. Alternatively, in other reputation-based approaches such as [7], [14]–[16], peers make peer selection decisions based on long-term local (or private) history of associate peers' upload behaviors. However, the focus of these systems is on maximizing *immediate* utility, which may be less desirable than maximizing *long-term* utility, as peers can repeatedly interact with each other in a long period of time.

In this chapter, we model the peer interactions in the BitTorrent-like network as a $stochastic - game$ [1] – A repeated interaction (i.e., reciprocating resources) between several participants (i.e., peers) in which the underlying state of the environment changes stochastically, and is dependent on the decisions of the participants. Stochastic games extend the single participant Markov decision process (MDP) [17] to include multiple participants whose actions all impact the resulting utility and next state. In our model, we explicitly consider the strategic behavior peers, which can observe partial historic information of associated peers statistical reciprocation behaviors, through which the peers can estimate the impact on their future rewards and then adopt their best response. The estimation of the impact on the expected future reward can be performed using different types of interactive learning [18]. We use reinforcement learning method [19], as it allows the peers to improve their peer selection strategy using only knowledge of their own past received payoffs, without knowing the complete reciprocation behaviors of the peers in the network. Thus, we propose to replace the peer selection policy with a reinforcement learning foresighted resource reciprocation policy. The resource reciprocation policy

is calculated by forecasting the impact of the current peer selection actions on the expected utility (i.e., future rewards) and maximizing it.

Thus, each peer can maximize its long-term utility based on the foresighted resource reciprocation policy. This can also provide an improved fairness, discourages free-riding, and enhances the system robustness. The foresighted resource reciprocation policy can replace both the tit-for-tat and the optimistic unchoke mechanisms in the regular BitTorrent protocol. In this chapter, we propose a BitTorrent-like protocol that applies the reinforcement learning foresighted strategy. Specifically, the protocol consists of three main processes:

- A learning process, which provides an updated information about statistical behaviors of the associated peers' resource reciprocation;
- A policy finding process, which computes the foresighted policy using the reinforcement-learning algorithm.
- A decision process, which determines the associated peers that will be unchoked and choked in every rechoke period based on the foresighted policy.

We implemented our proposed protocol on top of a BitTorrent client, and performed extensive experiments on a controlled PalnetLab testbed. We evaluate and quantify the performance of the proposed protocol, and compare its performance with the regular BitTorrent protocol. Based on the experimental results, the proposed protocol provides the following advantages against the regular BitTorrent protocol:

1) It improves the fairness. The peers that contribute more resources (i.e., higher upload capacities) can achieve higher download rates. However, the peers that contribute less resources may achieve limited download rates.
2) It promotes cooperation among high-capacity peers.
3) It discourages free-riding by limiting the upload to non-cooperative peers.
4) It improves the system robustness by minimizing the impact of free-riding on contributing peers' performance.

The rest of this paper is organized as follows. In Section II we briefly describe the BitTorrent. In Section III we briefly define the stochastic games and describe the reinforcement learning foresighted resource reciprocation strategy. Section IV presents the design of our foresighted resources reciprocation protocol. Details of our protocol implementation are discussed in Section V. The experiment results are presented in Section VI. Finally, we discuss related works in Section VII, and the conclusions are drawn in Section VIII.

II. BitTorrent Overview

In this section, we briefly overview the BitTorrent protocol. The BitTorrent protocol is a peer-to-peer content distribution protocol that scales efficiently with large number of participating clients.

Before the content distribution process begins, the content provider divides the possessed data content into multiple *pieces*, or *chunks*. Then, the provider creates a *metainfo file*, which contains information that is necessary to initiate the content downloading process. The metainfo file includes the address of the *tracker*, a coordinator which facilitates peer discovery.

A client downloads the metainfo file before joining a *torrent* (or swarm) – a group of peers interested in a particular content. Then, it connects the tracker to receive a *peer set*, which consists of randomly selected peers currently exchanging the same content. The peer set may include both *leechers*, peers that are still downloading content pieces, and *seeds*, peers that have the entire content and upload it to others. The client can then connect and exchange (or, *reciprocate*) its content pieces with its *associated peers* – the peers in its peer set.

While reciprocating content pieces, each leecher determines a set of peers among its peer set that can download its content pieces. The peer selection is determined by *choking mechanisms* and represented by *choking decisions*. BitTorrent leechers apply two choking mechanisms: the *tit-for-tat* resource reciprocation mechanism, and the *optimistic unchoke* mechanism. The tit-for-tat mechanism prefers the peers that upload their data at the highest rate among the associated peers. Specifically, in every *rechoke period*, typically 10 seconds, a leecher checks the current download rates from its associated peers and selects the peers that are uploading their data at the highest rates. Then, the leecher uploads only to the selected associated peers, while choking the rest of them during the rechoke period. The upload amount is a function of the available bandwidth for the uploading process. This available bandwidth is divided equally among the unchoked peers. The optimistic unchoke mechanism reserves a portion of the available upload bandwidth in order to provide pieces to randomly selected peers. The purpose of this mechanism is to enable the leechers to continuously discover better partners, and bootstrap newly joining leechers into the tit-for-tat mechanism. Optimistic unchokes are randomly rotated among the associated peers, typically once every three rechoke periods, allowing enough time for leechers to demonstrate their cooperative behaviors. The number of unchoked peers (slots) may vary depending on specific implementation, and it might be fixed or changed dynamically as a function of the available upload bandwidth.

Seeds deploy different choking mechanism as they already completed to download content. The most common implementation is based on the round-robin that aims to distribute data uniformly. This implementation is applied in our experiments.

III. Foresighted Resource Reciprocation Strategy

Peers in BitTorrent-like systems have to make a repeated peer selection decision given their dynamically changing environment which they experience. The evolution of the peers' interactions across the various rechoke periods can be modeled as repeated stochastic interaction. Whereas the Markov Decision Process (MDP) is a decision problem for one participant (i.e., peer) in an (unknown) environment [20], when multiple participants interact with each other in such an environment, this becomes a stochastic game [1], [21] problem (i.e., n-participant MDP). The time is discrete in the

stochastic game, and at each time slot (i.e., rechoke period), every participant has its own state and its own action space for that state. Every time slot the participants choose their own actions independently and simultaneously. After that, the participants are rewarded and transit to the next states. The reward (received by each of the participants), and the state transition also is contingent upon other participants' states and actions. Specifically to our model, during the repeated multi-peer interaction, the peers can observe partial historic information of associated peers reciprocation behaviors, through which the peers can estimate the impact on their future rewards and then adopt their best response. The estimation of the impact on the expected future reward can be performed using different types of interactive learning [18]. Here, we use reinforcement learning method [19], as it allows the peers to improve their peer selection strategy using only knowledge of their own past received payoffs, without knowing the complete reciprocation behavior of the peers in the network. In this learning framework of stochastic-game, the learning peers attempt to maximize their expected rewards.

Formally, a stochastic game is a tuple, $\langle I, \mathbf{S}, \mathbf{A}, P, R\rangle$, where I is a set of participants (peers), i.e., $I = \langle 1, ..., M\rangle$, $\mathbf{S}$ is the set of state profiles of all peers, i.e., $\mathbf{S} = S_1 \times ... \times S_M$ with $\mathbf{S}_j$ being the state space of peer j, and $\mathbf{A}$ is the joint action space $\mathbf{A} = A_1 \times ... \times A_M$, with $\mathbf{A}_j$ being the action (peer selection) space for peer j. $P : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$ is a state transition probability function that maps from state profile $S(t) \in \mathbf{S}$ at time t into the next state profile $S(t+1) \in \mathbf{S}$ at time $t+1$ given corresponding joint actions $A(t) \in \mathbf{A}$. Note that t here is discrete and measured in time slots. Finally, $R : \mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}_+$ is a reward vector function defined as a mapping from the state profile $S(t) \in \mathbf{S}$ at time t, and corresponding joint actions $A(t) \in \mathbf{A}$ to an $M-$dimensional real vector with each element being the reward to a particular participant.

1) State Space $\mathbf{S}_j$: A state of peer j represents a set of resources received from the peers in C_j, where C_j denotes the set of peers associating with peer j. Thus, it may represent the uploading behavior of its associated peers, or equivalently, it can capture peer j's download rates from its associated peers. The upload rates from peer $i \in C_j$ to peer j at time t are denoted by $UL_{i,j}(t)$. In our proposed protocol, an uploading behavior of peer i observed by peer j is denoted by s_{ij}, and defined as

$$s_{ij} = \begin{cases} 1, & \text{if } UL_{i,j} > \theta_j \ , \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where θ_j is a pre-determined threshold of peer j. [1] Thus, s_{ij} can be expressed with one bit and the state space of peer j can be expressed as

$$\mathbf{S}_j = \{ (s_{1j}, \ldots, s_{Nj}) |\, s_{kj} \in \{0, 1\} \text{ for all } k \in C_j \}, \tag{2}$$

where N denotes the number of peer j's associated peers, i.e., $|C_j| = N$. Therefore, a state $S_j(t) \in \mathbf{S}_j$ can capture the uploading behavior of the associated peers at time t. A state can be described using N bits, and thus, the cardinality of the state space is 2^N.

[1] In order to minimize the computational complexity, we consider $s_{ij} \in \{0, 1\}$ in this paper. However, the granularity of state can be easily extended.

A. The Action Space of a Peer - $\mathbf{A}_j$

An action of peer j represents a set of its peer selection decisions. The peer selection decision of peer j to peer i at time t is denoted by a_{ji}, and is defined as

$$a_{ji}(t) = \begin{cases} 0, & \text{if peer } j \text{ chokes peer } i \ , \\ 1, & \text{otherwise,} \end{cases} \tag{3}$$

Thus, a_{ji} can be expressed with one bit. The action space of peer j can be expressed as

$$\mathbf{A}_j = \{ (a_{j1}, \ldots, a_{jN}) |\, a_{jk} \in \{0, 1\} \text{ for all } k \in C_j \}, \tag{4}$$

Hence, an action $A_j(t) \in \mathbf{A}_j$ is a vector that consists of peer j's peer selection decisions to its associated peers at time t. Thus, an action of peer j for its N associated peers can be described with N bits. In the proposed protocol, we assume that peer j is able to unchoke a limited number of peers, denoted by $N_u (\leq N)$, implying that the cardinality of the action space is $\binom{N}{N_u}$. Note that in order to reduce the complexity, peer j allocates the same amount of upload bandwidths to all unchoked peers. Thus, the bandwidth allocated to an unchoked peer i by peer j at time t is determined by $UL_{j,i}(t) = B_j/N_u$, where B_j is the maximum upload bandwidth available to peer j.

B. State Transition Probability in a Peer

A state transition probability represents the probability that an action $A_j(t) \in \mathbf{A}_j$ of peer j in state $S_j(t) \in \mathbf{S}_j$ at time t will lead to another state $S_j(t+1) \in \mathbf{S}_j$ at $t+1$. Thus,

$$P_{A_j(t)}(S_j(t), S_j(t+1)) = \Pr(S_j(t+1) | S_j(t), A_j(t)). \tag{5}$$

The state transition probability functions can be estimated at a particular peer j based on the history of $S_j(t)$, $A_j(t)$ and $S_j(t+1)$, which may be stored in a transition table. Thus, the transition table size is in order of $O(\binom{N}{N_u} \cdot 2^N \cdot 2^N)$. While we deploy an empirical frequency based algorithm to estimate the state transition probability function, which is presented in Section IV-A, other algorithms (e.g., [22]) can also be used.

C. The Reward of a Peer - R_j

The reward of a peer in a state is its total estimated download rate in the state. Thus, a reward of a peer in a state is the sum of the estimated download rates from all its associated peers. More specifically, a reward of peer j from state $S_j \in \mathbf{S}_j$ can be expressed as

$$R_j(S_j) = \left\langle S_j, [UL_{i,j}]_{i \in C_j} \right\rangle \tag{6}$$

where $\langle \cdot \rangle$ denotes the inner-product operation.

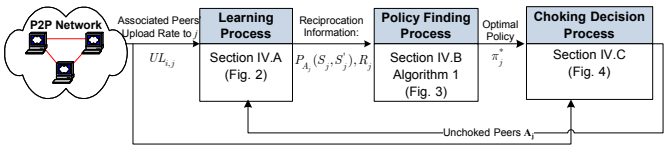

Fig. 1. The Main Processes in Proposed Protocol Design

1) Resource Reciprocation Policy π_j^*: We define a history of the stochastic game as $h^t = \{S^0, A^0, R^0, ..., S^{t-1}, A^{t-1}, R^{t-1}\} \in \mathcal{H}^t$, which summarizes all states, actions and rewards of the peers in the network up to time $t-1$, where $\mathcal{H}^t$ is the set of all possible histories up to time t. Nevertheless, during the stochastic game, each peer j cannot observe the entire history, but rather a portion of the history. The observation of peer j is denoted as $o_j^t \in \mathcal{O}_j^t$ and $o_j^t \subset h^t$. Note that the current state s_j^t can be always observed, i.e. $s_j^t \in o_j^t$. Thus a peer selection policy $\pi_j^t : \mathcal{O}_j^t \rightarrow \mathbf{A}_j$ for peer j at the time t is defined as a mapping from the observations up to the time t into the specific action, i.e. $a_j^t = \pi_j^t(o_j^t)$. Furthermore, a policy profile π_j for peer j aggregates the peer selection policies over the entire course of the stochastic game, i.e. $\pi_j = (\pi_j^0, ..., \pi_j^t, ...)$. The policy profile for all the peers at time slot t is denoted by $\pi^t = (\pi_1^t, ..., \pi_M^t) = (\pi_j^t, \pi_{-j}^t)$.

The policy of peer j is calculated using reinforcement-learning algorithm that maximizes the cumulative discounted expected reward. The expected reward is defined for a peer j in state $S_j(t)$ at time $t = t_c$ given a discount factor γ_j as

$$R_j^f(S_j(t_c)) \triangleq \sum_{t=t_c+1}^{\infty} \gamma_j^{(t-(t_c+1))} \cdot R_j(S_j(t)). \quad (7)$$

The policy π_j maps each state $S_j(t) \in \mathbf{S}_j$ into an action, i.e., $\pi_j(S_j) = A_j(t)$ such that each action maximizes $R_j^f(S_j(t_c))$. The policy can be deployed as a peer selection algorithm, such that each peer can maximize its long-term utility. While the policy π_j can be obtained using well-known methods such as value iteration and policy iteration [17], these algorithms may require very high computational complexity if the number of associated peers is significantly large. Hence, it is important to reduce the computational complexity of the policy calculation, such that the reinforcement learning foresighted strategy is deployed in practice.

IV. The Protocol Design

In this section, we describe the proposed protocol design that replaces the peer selection tit-for-tat and optimistic unchoke mechanisms deployed in regular BitTorrent systems with a foresighted resource reciprocation mechanism.

The protocol design is summarized in Fig. 1. The protocol consists of three main processes running in parallel: (1) *the learning process*, which provides an updated information about statistical behaviors of the associated peers' resource reciprocation; (2) *the policy finding process*, which computes the foresighted resource reciprocation policy; (3) and *the decision process*, which determines the peer selection decisions during the course of each rechoke period. More details about these processes are discussed next.

A. The Learning Process

As discussed in Section III, in order to find the foresighted resource reciprocation policy, each peer needs to know other peers' states, their rewards, and their state transition probabilities in order to derive its own optimal policy. However, a peer cannot exactly know the other peers information, due to information that is kept private, network scalability constraints, the time-varying network dynamic, and more. Thus, to improve the peer selection policy, a peer can only predict the impacts of dynamics(uncertainties) caused by the competing peers based on its observations from the past. Thus each peer needs to update the above information regularly through the learning process, while downloading content from its associated peers.

The learning process consists of two main methods (see Fig. 2) that compute the estimated reward and state transition probability.

1) Reward Calculation: The reward of peer j represents its download rates from its associated peers (or equivalently, the total upload rates of its associated peers) estimated by peer j. In the rewards calculation method, the associated peers are classified into two types based on the available information about their resource reciprocation history.

For associated peers that have reciprocated their resources with peer j, referred to as *peers with reciprocation history*, peer j estimates their upload rates based on weighted average of the past upload rate samples. This can reduce the fluctuation induced by the protocol and network dynamics in the sampled upload rates of the associated peers. Specifically, peer j estimates the upload rates $\hat{r}_{i,j}$ of peer $i \in C_j$ based on recently observed resource reciprocation $UL_{i,j}$ as

$$\hat{r}_{i,j} \leftarrow \alpha_j \cdot UL_{i,j} + (1-\alpha_j)\hat{r}_{i,j} \quad (8)$$

where α_j denotes the weight for most recent resource reciprocation.

For associated peers whom have *not* reciprocated their resources with peer j, referred to as *peers without resource reciprocation history*, peer j assumes their upload rates. Peer j optimistically initiates the information about such peers by assuming that they reciprocate their resources with high probability and high upload rate. This enables peer j to efficiently discover additional peers, and bootstrap newly joining peers, which is important for the efficiency of the system. Whenever peer j uploads to a peer without resource reciprocation history and the peer do not upload to j in return, peer j reduces the peer's presumed upload rate, as this provides j with more confidence that the particular peer may not actively reciprocate its data. This also prevents the associated peers from taking advantage through optimistic initialization and possible free-riding.

2) State Transition Probability Calculation: The state transition probabilities are updated every rechoke period, and thus, each peer can capture the time-varying resource reciprocation

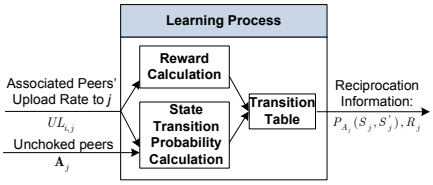

Fig. 2. The Learning Process

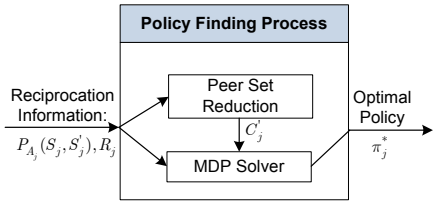

Fig. 3. The Policy Finding Process

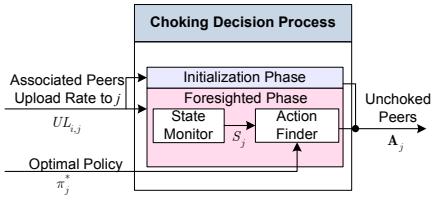

Fig. 4. The Decision Process

behaviors of its associated peers. Every rechoke period at $t+1$, peer j stores a 3-bit triplet $(S_j(t), A_j(t), S_j(t+1))$. Peer j stores the triplets to the associated peers that are in a particular peer set referred to as reduce peer set, which will be discuss later in this section or peers that uploaded to peer j at time t or $t+1$. In this paper, we assume that the state transition probability functions are computed based on the empirical frequency, and the state transition of each peer is independent. Thus, the probability, $\Pr(s_{ij}, a_{ji}, s'_{ij})$, where $s_{ij}, s'_{ij} \in \mathbf{S}_j$, can be expressed as

$$P_{A_j(t)}(S_j(t), S_j(t+1)) = \prod_{i=1}^{N} \Pr(s_{ij}(t+1)|s_{ij}(t), a_{ji}(t))$$

B. The Policy Finding Process

The policy finding process runs in parallel with the learning process, while computing the foresighted resource reciprocation policies based on the information obtained from the learning process. This process is depicted in Fig. 3.

Finding the foresighted policy by solving the MDP may require significantly high computational complexity if the number of the associated peers becomes large. Hence, for practical implementation of the foresighted resource reciprocation mechanism, it is critical to reduce the number of peers that needs to be considered (see Section III). Thus, this process begins with reducing the set of associated peers, and then, finds the foresighted resource reciprocation policy π_j by maximizing the cumulative discounted expected reward (Equation 7) in the reduced peer set.

1) Reducing Associated Peer Set: As discussed, it is important for peer j to reduce the set of associated peers to find π_j, while keeping the peers that reciprocate their resources with higher probability and with higher upload rate in the reduced peer set. Specifically, peer j computes the expected rewards $K_{i,j}$ from each peer $i \in C_j$, defined as

$$K_{i,j} = R_{i,j} \times Pr(i,j), \tag{9}$$

where $R_{i,j}$ and $Pr(i,j)$ denote the estimated reward from peer i and the probability of resource reciprocation with peer i, respectively. Based on computed $K_{i,j}$, peer j reduces its associated peer set by iteratively consider eliminating the peers with the smallest $K_{i,j}$ in its associated peer set. The algorithm for peer set reduction is presented in Algorithm 1. The algorithm computes $K_{i,j}$ in (9) for $i \in C_j$ (lines 3,4). Then, the associated peers are ordered based on computed $K_{i,j}$ (line 5). The peer set reduction is performed in while loop (lines 7- 18) that reduces the peer set by c_2 peers in every iteration. In the loop, the algorithm selects c_1 peers with the smallest $K_{i,j}$ values denoted by G (line 8), from the reduced group of peers C'_j. It then obtains policy $\pi_{j,G}$ for the peers in G. (line 9). Based on $\pi_{j,G}$, it calculates the probabilities for the peers to be unchoked (lines 10-14). Given the calculated probability, it removes the c_2 peers with the lowest probability to be unchoked (18). The algorithm runs until $|C'_j| = T$ (line 7).

2) Scaling: Scaling of the rewards is considered in cases, when the number of reciprocation samples is small in comparison to the difference between the highest and the lowest upload rates that are expressed in the P2P network.

Algorithm 1 Peer-Set Reduction Algorithm

1: **INPUT :**
 · C_j - set of associated peers of peer j
 · T - target output set size (constant)
 · $R_{i,j}$ - estimated rewards of peer i
 · $P(S_j)$ - the probability to be in state S_j
 · $Pr(i,j)$ the resource reciprocation probability of peer i
 · c_1, c_2 - constants such that $T \gg c_1 > c_2$
2: **OUTPUT :** A reduced set of peers $C'_j \subseteq C_j$ s.t. $|C'_j| = T$
3: **for all** $i \in C_j$ **do**
4: $K_{i,j} = R_{i,j} \times Pr(i,j)$;
5: order C_j in a non-decreasing order of the $K_{i,j}$;
6: $C'_j = C_j$;
7: **while** $|C'_j| > T$ **do**
8: G = $\left\langle C'_{j\,1}, ..., C'_{j\,c_1} \right\rangle$;
9: calculate $\pi^*_{j,G}$ the optimal policy for the set G;
10: **for all** $i \in G$ **do**
11: $Pu_i = 0$
//Calculate Pu_i, the probability
//that j unchokes i using $\pi^*_{j,G}$ policy;
12: **for all** $s_j \in S_j$ **do**
13: **if** $\pi^*_{j,G}(s_{(i,j)}) = 1$ **then**
14: $Pu_i = Pu_i + P(s_j)$;
15: order G in a non-decreasing order of the Pu_i values;
16: **if** $c_2 > |C'_j| - T$ **then**
17: $c_2 = |C'_j| - T$;
18: $C'_j \leftarrow C'_j - < G_1, ..., G_{c_2} >$;
19: **return** C'_j

C. The Decision Process

The decision process includes two phases: the initialization phase and the foresighted phase (see Fig. 4).

1) Initialization Phase: Since no information about associated peers is available for a newly joined peer j, peer j begins with adopting the regular BitTorrent mechanisms (i.e., the tit-for-tat mechanism and the optimistic unchoke mechanism) in the initialization phase. This enables the peer to collect information such as the rewards and state transition probabilities with respect to its associated peers. During this phase, j discovers new peers, i.e., downloads from peers for the first time. Once j's peer discovery is slowed down (see Section V for more details), it replaces the regular BitTorrent mechanisms with the foresighted resource reciprocation mechanism, and operates in the foresighted phase.

2) Foresighted Phase: Once the foresighted resource reciprocation policy is available, peer j determines the peer selection decisions based on the foresighted policy obtained from the policy finding process in every rechoke period. Peer j first determines its current state S_j and then finds an optimal action A_j mapped by the policy π_j, i.e., $A_j = \pi_j(S_j)$. A_j is a set of peers that peer j unchokes.

V. Implementation

In this section, we discuss the implementation of the foresighted resource reciprocation protocol prototype and study how to determine several design parameters.

Our P2P client is implemented based on Enhanced CTorrent client, version 3.2 [23]. We enhances the original client such that our client can operate in *foresighted mode*, where it reciprocates its resources based on the proposed foresighted resource reciprocation mechanism, or in *regular mode*, where it reciprocates its resources based on the regular BitTorrent peer selection mechanisms. We add functionality for the foresighted mode to maintain the new protocol requests. More specifically, we implemented the three different processes that are discussed in Section IV.

A. The Learning Process

The learning process consists of two methods, the reward calculation method and the state transition probability calculation method. We now discuss the reward calculation method.

1) Reward Calculation Method: The reward calculation method can be applied differently depending on the associated peer types: peers with or without reciprocation history.

While calculating the reward of a peer with resource reciprocation history, obviously the samples of $UL_{i,j}$ will fluctuate over rechoke period time due to P2P network dynamics. Because of this fluctuation, $UL_{i,j}$ samples may be atypical. Thus, a typical upload rate of a peer with reciprocation history can be estimated based on weighted average of the samples as in (8). This is the estimated reward of peer j obtained from peer i. As recent resource reciprocation is considered more important than the previous reciprocations, $\alpha_j > 0.5$. Based on several trials for α such that $0.5+\epsilon \leq \alpha \leq 1-\epsilon$ for small $\epsilon > 0$, on various sets of our experiments (see more details in Section VI), we can verify that the smallest α achieves less fluctuation of the reward. Thus, we set α to $0.5 + \epsilon$ where $\epsilon = \frac{1}{16}$. Fig. 5 shows the sampled upload rates $U_{i,j}$ of a peer i having 9KB/sec upload bandwidth (that simultaneously uploads to 4 peers) and the correspondingly estimated rewards $\hat{r}_{i,j}$ as measured by another peer in the network. Clearly, we can observe less variations of the $UL_{i,j}$ in the computation of the $\hat{r}_{i,j}$.

For a peer i without reciprocation history, a leecher j optimistically initializes the information about the rewards and the reciprocation probabilities of its associated peers. Specifically, the initial estimated upload rate is set to be the highest upload rate $R_{i,j}^{max}$ that is pre-determined in the P2P network, i.e., $R_{i,j} \leftarrow R_{i,j}^{max}$, and the probability of reciprocation with j is initiated to 1, i.e., $Pr(i,j) \leftarrow 1$. This optimistic initialization enables newly joined leechers to download almost immediately. Peer j needs to keep updating the initially assumed reward in every non-reciprocated event (i.e., peer j uploads resources to peer i while peer i does not upload resources to peer j). When peer j estimates the reward for peer i, peer j can assume that (i) $\hat{r}_{i,j}$ satisfies

$$\frac{\hat{r}_{i,j}(n-1)}{\hat{r}_{i,j}(n)} < \frac{\hat{r}_{i,j}(n)}{\hat{r}_{i,j}(n+1)}, \tag{10}$$

where n denotes the number of non-reciprocated events. This means that the ratio of the estimated rate of two consecutive events is an increasing function of n. This implies the increasing uncertainty about peer i's reciprocation behavior. Moreover, (ii) $\hat{r}_{i,j}(n)$ decreases exponentially such that it approaches 0 after several attempts, in order to prevent the non-reciprocated behavior including free-riding. Thus, a function satisfying (i) and (ii) can have a form, such as

$$r(n) = \beta^{g(n)} \times R_{i,j}^{max}, \tag{11}$$

where $\beta(<1)$ is a constant and $g(n) > 1, \forall n \geq 1$ is a function that grows faster than a linear function. In our implementation, we use function $r(n) = 0.95^{2^n} \times R_{i,j}^{max}$, as the function satisfies properties (i) and (ii) as shown in Fig. 6.

B. The Policy Finding Process

As shown in Section IV, in every iteration of the policy finding process, the associated peer set is first reduced. Based on our experiments, we observe that when the reduced size of peer set is more than 7 peers, finding the foresighted strategy requires significant computational complexity. Thus, in our implementation, we set the size of the reduced peer set to 7, i.e., $T = 7$ in Algorithm 1.

The policy is calculated and holds for up to additional three rechoke periods, which is determined based on the tradeoff between the time for enough reciprocation and the time for capturing the network dynamics.

C. The Decision Process

The initialization phase and the foresighted phase in the decision process are implemented as follows.

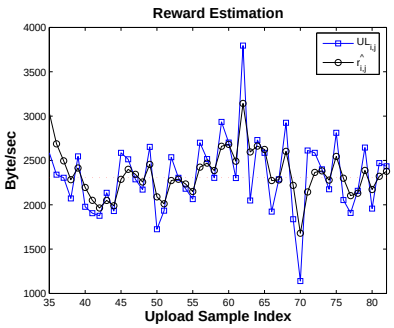

Fig. 5. Upload Samples and Reward Estimations

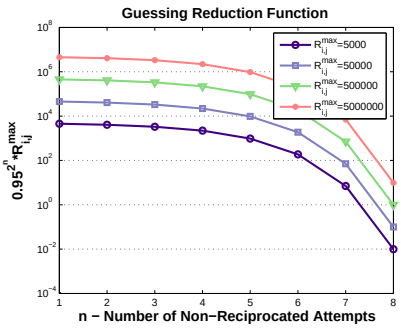

Fig. 6. Guessing Reduction Function

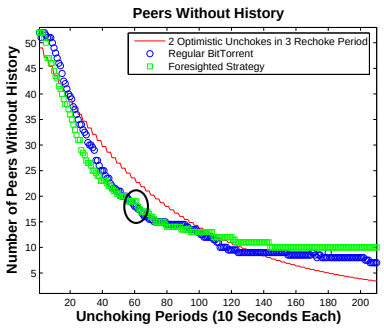

Fig. 7. Discovering Rate of Peers Without History

- *Initialization Phase:* In the initialization phase, peer j makes peer selection decisions based on the regular BitTorrent mechanisms, as it does not have enough information to calculate the foresighted policy. The leechers determine the duration of the initialization phase individually. We study extensive experiment results, which include both flash crowd scenarios as well as steady state scenarios. In these experiments, the number of peers that have not uploaded to peer j from the beginning of the downloading process is counted every rechoke period. Fig. 7 shows the median of the counted numbers of peers collected from all the leechers in the network over time (rechoke periods) for several experiments of flash-crowd scenarios. The Figure shows that the peer counted value is exponentially decreasing and stabilized fast. Then, peer j can switch from the initialization phase to the foresighted phase. In our implementation, a peer j counts the number of peers without reciprocation history within every rechoke period. Once the count reduces by one in duration of three rechoke periods and for two consecutive durations (i.e., six rechoke periods), peer j switches to the continuous phase and starts to adopt the foresighted strategy. Based on our experiments, peers switch from initialization phase to continuous phase approximately 60 rechoke periods later in the flash-crowd scenarios and approximately 36 rechoke periods later in the steady-state scenarios. However, different network settings might lead to different durations of the initialization phase.

- *Foresighted Phase:*

In the foresighted phase, the choking decisions are made based on the foresighted policy, in every 10 seconds (as in regular BitTorrent). The selected peers will be unchoked for the entire 10 seconds rechoke period. The minimum number of unchoked peers is 4. The number of unchoked peers can increase if (1) the peer that makes the peer selection decision does not saturate its upload capacity, or, (2) the upload bandwidth of the peer that makes the peer selection decision is higher in comparison to most of the peers it interacts with.

We compare the performance of our protocol to the performance of the regular BitTorrent implemented with the Enhanced CTorrent code. The minimum number of unchocke slots in the regular BitTorrent implementation is 4. The number of slots can increase if a peer's upload capacity is not saturated. In this implementation, one unchoke slot is always reserved for optimistic unchokes that are rotated every three rechoke periods.

VI. Experimental Evaluation

We perform extensive experiments on a controlled testbed, in order to evaluate the properties of the proposed protocol.

A. Methodology

All of our experiments are performed on the PlanetLab experimental platform [24], which utilizes the nodes (machines) located across the globe. We execute all the experiments consecutively in time on the same set of nodes. Unless otherwise specified, the default implementations of leecher and seed in regular BitTorrent systems are deployed.

The upload capacities of the nodes are artificially set according to the bandwidth distribution of typical BitTorrent leechers [3]. The distribution was estimated based on empirical measurements of BitTorrent swarms including more than 300,000 unique BitTorrent IPs. Since several nodes are not capable to match the target upload capacities determined by the bandwidth distribution, we scale the upload capacity and other relevant experimental parameters such as file size by 1/20th. However, we have not set limitation on download bandwidth.

All peers start the download process simultaneously, which emulates a flash crowd scenario. The initial seeds are stayed connected through the whole experiments. To provide synthetic churn with constant capacity, leechers disconnect immediately after completion of downloading the entire file, and reconnect immediately while requesting the entire file again. This enables our experiments to have the same upload bandwidth distribution during the entire experiment time.

Unless otherwise specified, our experiments host 54 PlanetLab nodes, 50 leechers and 4 seeds with combined capacity of 128 KB/s serving a 100 MB file.

B. Experiment Results: Performance of Leechers in Network without Free-Riders

We compare a system consisting of all leechers adopting the regular BitTorrent protocol, to a system consisting of all leechers adopting the proposed protocol based on the foresighted strategy. In this section, we assume that there is no free-rider in the networks. Fig. 8 shows the download

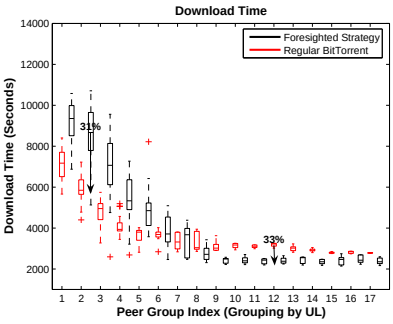

Fig. 8. Download completion time for leechers

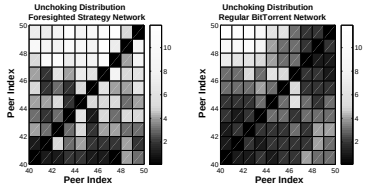

Fig. 9. Unchokes among the 20% fastest peers

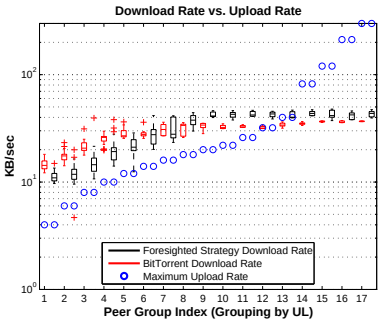

Fig. 10. download rate vs. upload rate

completion time of leechers. For each group of leechers having the same upload capacity, separate boxplots are depicted [25] for the different scenarios. The top and the bottom of the boxes represent the 75th and the 25th percentile sample of download time, respectively, over all 5 runs of the experiments. The markers inside the boxes represent the median, while the vertical lines extending above and below the boxes represent the maximum and minimum of samples of download time within the ranges of 1.5 time the box height from the box boarder. Outliers are marked individually with "+" mark.

TThe results show the clear performance difference among high-capacity leechers, which are the fastest 20% leechers, and low-capacity leechers, which are the slowest 80% leechers. High-capacity leechers can significantly improve their download completion time – leechers having the upload capacity of at least 18kB/sec improve their download completion time by up to 33% in median. Unlike in the regular BitTorrent system, where leechers determine their peer selection decisions based on the myopic tit-for-tat that uses only the last reciprocation history, the leechers adopting the foresighted strategy determine their peer selection decisions based on the long-term history. This enables the leechers to estimate the behaviors of their associated peers more accurately. Moreover, since part of the peer selection decisions is randomly determined in the regular BitTorrent, there is a high probability that high-capacity leechers need to reciprocate with the low-capacity leechers [3]. However, the randomly determined peer selection decisions are significantly reduced in the proposed approach, as the random decisions are taken only in the initialization phase or in order to collect the reciprocation history of newly joined peers. As a result, the high-capacity leechers increase the probability to reciprocate resources with the other high-capacity leechers.

This is confirmed in the results of Fig. 9, which shows the unchoking percentage among the 20% high-capacity leechers, comparing the two different systems. It is clearly observed that the collaboration among high-capacity leechers improves when leechers adopt the foresighted strategy. Thus, we can conclude that the foresighted strategy improves the incentive mechanisms in BitTorrent networks: as a leecher contributes more to the network, it achieves higher download rate.

Recent studies [3]–[5], [15] show that the regular BitTorrent

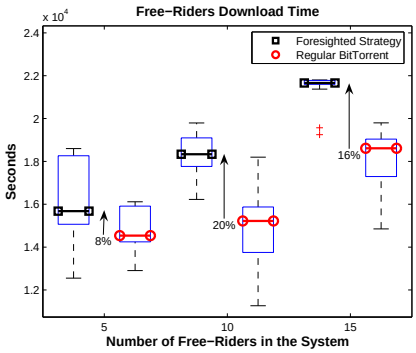

Fig. 11. Download completion time for free-riders

protocol suffers from unfairness particularly for high capacity leechers. In Fig. 10, we compare the upload rates and the average download rates of the leechers. The ratio of these values can indicate the degree of fairness in the system. The results in Fig. 10 show that fairness is improved as the leechers adopt the foresighted strategy, since high-capacity leechers increased their download rate getting closer to their upload rate, in spite of the restriction of limited seeds' upload rate. On the other hand, in the system where leecher adopts the foresighted strategy, the download rates of low-capacity leechers decrease, getting close by at most 36% to their upload rates, compare to the regular BitTorrent system. However, all the peers that are slowed down by the foresighted strategy still download faster than their upload rate.

C. Experiment Results: Performance of Leechers in Network with Free-Riders

In this section, we investigate how effectively the proposed protocol can prevent selfish behaviors such as free-ridings. Note that the foresighted strategy shows a similar performance for the leechers that upload their content in the network that includes free-riders (i.e., shows the improved fairness, etc.). Hence, in this section, our focus is on studying how the free-riders are punished due to their selfish behaviors. Fig. 11 shows the time that the free-riders complete downloading 100MB file in a network consists of 50 contributing leechers, and increasing number of free-riders (i.e., 5, 10, and 15 free-riders). It compares the results of the foresighted strategy

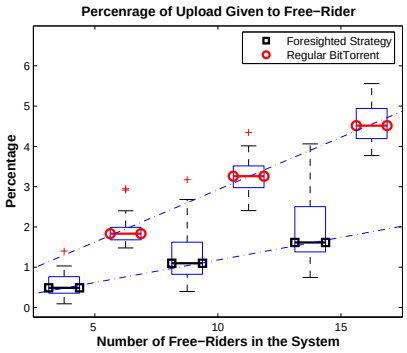

Fig. 12. Percentage of free-riders' download from contributing leechers

system to the regular BitTorrent system. The Figure confirms that the foresighted strategy enables the leechers to effectively penalize the free-riders, as it takes longer time for the free-riders to complete their downloads (requires 8%-20% more time as measured by the median, in comparison to the regular BitTorrent protocol).

When leechers adopt the foresighted strategy, they can efficiently capture the selfish behavior of the free-riders. Thus, they unchoke the free-riders with a significantly lower probability. Hence, the free-riders can download their content mainly from seeds not from the leechers. The results shown in Fig. 12 also confirm that the leechers in the regular BitTorrent system upload approximately 2.8-3.7 times more data to the free-riders compared to the leechers in the system where foresighted strategy is adopted. This also shows that the P2P networks consist of the leechers adopting the foresighted strategy are more robust to the selfish behaviors of peers than the networks operating with the regular BitTorrent protocol. For example, in the network with 15 free-riders, the leechers in the regular BitTorrent systems upload 4.5% of their total upload capacity to free-riders, while they only upload 1.6% of their upload capacity in the foresighted strategy system.

Therefore, our experiment results confirm that the foresighted strategy provides more incentives for leechers to maximize their upload rate by enabling the leechers to discourage non-cooperative behaviors such as free-riding, improves fairness, and enhances the robustness of the network.

VII. Related Work

Extensive research has focused on modeling and analyzing the performance of the BitTorrent systems, since the main mechanisms and the design rationale of the BitTorrent protocol first described [2].

Qiu and Srikant [26] studied a fluid analytical model of BitTorrent systems. They analytically studied the choking mechanism and how it affects the peer performance. They showed that the optimistic unchoke mechanism may allow free-riding. They also claimed that the system with tit-for-tat strategy eventually converges to a Nash equilibrium where fairness is achieved and all peers download at their upload capacities. However, as shown in our results, which are in consistent with other existing works such as [3], [5], [15], [27] the choking mechanism in BitTorrent may fail to attain fairness for a realistic swarms. Fan *et al.* [28] characterized the design space of BitTorrent-like protocols capturing the fundamental tradeoff between performance and fairness. We also study such tradeoff and show that the foresighted strategy improves the fairness in the system for the cost of reduced download rates of low-capacity leechers. This encourages leechers to contribute more resources (i.e., maximize their upload rate).

Other researchers have studied the feasibility of free-riding behavior, Shneidman *et al.* [29] showed that it is possible to free-ride in BitTorrent systems. They identified forms of strategic manipulation that are based on Sybil attacks and uploading garbage data. Liogkas *et al.* [8] implemented three exploits that allow free-riders to obtain higher download rates under specific circumstances. Locher *et al.* [9] with BitThief extended this work by showing that free-riders can achieve higher download rate, even in the absence of seeds. Similarly, Sirivianos *et al.* [10] showed that free-rider that can maintain a larger-than-normal view of the system has much higher probability to receive data from seeds and via optimistic unchoke. Our protocol replaces the optimistic unchokes, the most important vulnerability identified in these studies, with the foresighted policy based unchokes.

Fairness in BitTorrent systems was studied as well. Geo *et al.* [5] showed the lack of fairness in BitTorrent systems. Piatek *et al.* [3], observed the presence of significant altruism in BitTorrent, where peers make contributions that do not directly improve their performance. They proposed a new choking mechanism that reallocates upload bandwidth to maximize peers' download rates. Izhak-Ratzin in [15] identified the potential of significant different between leecher's upload and download rates and proposed the Buddy protocol that matches peers with similar bandwidth. Legout *et al.* [6] studied clustering of peers having similar upload bandwidth. They observed that when the seed is underprovisioned, all peers tend to complete their downloads approximately at the same time, regardless of their upload rates. Moreover, high-capacity peers assist the seed to disseminate data to low-capacity peers. This can happen because the tit-for-tat strategy is based on short-term history. A peer can benefit from the tit-for-tat strategy only if it can continuously upload pieces and as long as it receives pieces of interest in return. Piatek *et al.* [7] showed that this is not always possible as peers can have no piece to offer. Our work also considers the unfairness in BitTorrent systems, and shows that the proposed approach can improve the fairness by using a long-term history based strategy.

In order to reduce free-riding and encourage collaboration, various reputation systems have been proposed. Payment systems (e.g., [30], [31]) which enables peers to earn credits according to their uploads to other peers have been proposed . However, in practice these systems require a centralized entity to prevent cheating, and thus, they have arguably scalability limitation. To overcome such weaknesses in payment systems, various design of reputation systems has been proposed (e.g., [7], [11]–[13], [32]). In these systems, peers can make choking

decisions base on private history as well as globally shared history. However, these reputation systems require significant communication overheads to maintain the global history. Moreover, there is no guarantee that each peer expresses the same behavior to different peers with different attributes. In addition, all these systems aim to maximize the immediate utility but not the long-term utility, which can show only suboptimal performance. To the best of our knowledge, we are the first to propose the foresighted strategy that can replace the existing mechanisms deployed in BitTorrent protocol, while maximizing long-term utility of participating leechers.

Finally, the theoretical aspects of the MDP-based foresighted strategy are discussed in our prior work [22]. These results motivate us to design the resource reciprocation mechanism that replaces the tit-for-tat and optimistic unchoke mechanisms that exist in BitTorrent.

VIII. Conclusion

In this paper, we propose the foresighted resource reciprocation mechanism to replace the tit-for-tat and the optimistic unchoke mechanisms in BitTorrent. The proposed strategy enables the peers to utilize the reciprocation history, such that they can efficiently capture the behaviors of their associated peers. Hence, the peers can improve their performance. Our experiment results show that the proposed protocol improves the fairness, thus provides more incentives to collaborate, and enhances the robustness of the P2P networks by effectively discouraging free-riding.

References

[1] D. Fudenberg and D. K. Levine, "The theory of learning in games," Master's thesis, Cambridge, MA: MIT Press, 1999.

[2] B. Cohen, "Incentives Build Robustness in BitTorrent," in *P2PEcon'03*.

[3] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in BitTorrent?" in *NSDI'07*.

[4] A. Bharambe and C. Herley and V. Padmanabhan, "Analyzing and improving a bittorrent network's performance mechanisms," in *INFOCOM*, 2006.

[5] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis, and modeling of BitTorrent-like systems," in *IMC05*.

[6] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and Sharing Incentives in BitTorrent Systems," in *SIGMETRICS'07*.

[7] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson, "One hop reputations for peer to peer file sharing workloads," in *NSDI*, 2008.

[8] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang, "Exploiting BitTorrent For Fun (But Not Profit)," in *IPTPS'06*.

[9] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, "Free Riding in BitTorrent is Cheap," in *HotNets-V (2006)*.

[10] M. Sirivianos, J. H. Park, R. Chen, and X. Yang, "Free-riding in BitTorrent Networks with the Large View Exploit," in *IPTPS'07*.

[11] S. Buchegger and J.-Y. le Boudec, "A Robust Reputation System for P2P and Mobile Ad-hoc Networks," in *Economics of P2P Systems*, 2004.

[12] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities." in *IEEE Transactions on Knowledge and Data Engineering, 16(7).*, 2004.

[13] M. Yang, Z. Zhang, X. Li, and Y. Dai, "An Empirical Study of Free-Riding Behavior in the Maze P2P File-Sharing System." in *IPTPS*, 2005.

[14] http://www.edonkey2000.com/, "edonkey."

[15] Rafit Izhak-Razin, "Collaboration in bittorrent systems," in *Networking 2009*.

[16] Rafit Izhak-Razin, Nikitas Liogkas, and Rupak Majumdar, "Team incentives in bittorrent systems," in *ICCCN 2009*.

[17] D. P. Bertsekas, *Dynamic Programming and Stochastic Control*. Academic P, 1976.

[18] M. Bowling and M. Veloso, "Rational and convergent learning in stochastic games," Master's thesis, Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), 2001.

[19] J. Hu and P. Wellman, "Multiagent reinforcement learning: theorectical framework and an algorithm."

[20] R. G. Gallager, "Discrete stochastic processes," 1996.

[21] L. S. Shapley, "Stochastic games," vol. 39, 1095-1100, Proceedings of the National Academy of Sciences of the United States of America, 1953.

[22] H. Park and M. van der Schaar, "A framework for foresighted resource reciprocation in P2P networks," *IEEE Trans. Multimedia*, vol. 11, no. 1, pp. 101–116, Jan. 2009.

[23] http://www.rahul.net/dholmes/ctorrent, "Enhanced CTorrent."

[24] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating System Support for Planetary-Scale Network Services," in *NSDI'04*.

[25] J. W. T. Robert McGill and W. A. Larsen, "Variations of box plots," *The American Statistician*, vol. 32, pp. 12–16, 1978.

[26] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," in *SIGCOMM'04*.

[27] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee, "Bittorrent is an auction: analyzing and improving bittorrent's incentives," *Sigcomm*, 2008.

[28] B. Fan, D.-M. Chiu, and J. C. Lui, "The Delicate Tradeoffs in BitTorrent-like File Sharing Protocol Design," in *ICNP*, 2006.

[29] J. Shneidman and D. C. Parkes, "Rationality and Self-Interest in Peer to Peer Networks," in *IPTPS 2003*.

[30] B. Wilcox-O'Hearn, "Experiences Deploying A Large-Scale Emergent Network," in *IPTPS*, 2002.

[31] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing." in *P2PEcon*, 2003.

[32] Qiao Lian, Yu Peng, Mao Yang, Zheng Zhang, Yafei Dai, and Xiaoming Li , "Robust incentives via multi-level tit-for-tat," in *IPTPS*, 2006.