Robust Incentive Techniques for Peer-to-Peer Networks

Michal Feldman[1] mfeldman@sims.berkeley.edu

Kevin Lai[2] klai@hp.com

Ion Stoica[3] istoica@cs.berkeley.edu

John Chuang[1] chuang@sims.berkeley.edu

[1]School of Information Management and Systems U.C. Berkeley

[2]HP Labs

[3]Computer Science Division U.C. Berkeley

ABSTRACT

Lack of cooperation (free riding) is one of the key problems that confronts today's P2P systems. What makes this problem particularly difficult is the unique set of challenges that P2P systems pose: large populations, high turnover, asymmetry of interest, collusion, zero-cost identities, and traitors. To tackle these challenges we model the P2P system using the Generalized Prisoner's Dilemma (GPD), and propose the Reciprocative decision function as the basis of a family of incentives techniques. These techniques are fully distributed and include: discriminating server selection, maxflow-based subjective reputation, and adaptive stranger policies. Through simulation, we show that these techniques can drive a system of strategic users to nearly optimal levels of cooperation.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; J.4 [**Social And Behavioral Sciences**]: Economics

General Terms

Design, Economics

Keywords

Incentives, peer-to-peer, free-riding, reputation, collusion, cheap pseudonyms, whitewash, prisoners dilemma

1. INTRODUCTION

Many peer-to-peer (P2P) systems rely on cooperation among self-interested users. For example, in a file-sharing system, overall download latency and failure rate increase when users do not share their resources [3]. In a wireless ad-hoc network, overall packet latency and loss rate increase when nodes refuse to forward packets on behalf of others [26]. Further examples are file preservation [25], discussion boards [17], online auctions [16], and overlay routing [6]. In many of these systems, users have natural disincentives to cooperate because cooperation consumes their own resources and may degrade their own performance. As a result, each user's attempt to maximize her own utility effectively lowers the overall utility of the system. Avoiding this "tragedy of the commons" [18] requires incentives for cooperation.

EC'04, May 17–20, 2004, New York, New York, USA.

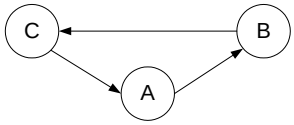

Figure 1: Example of asymmetry of interest. A wants service from B, B wants service form C, and C wants service from A.

We adopt a game-theoretic approach in addressing this problem. In particular, we use a prisoners' dilemma model to capture the essential tension between individual and social utility, asymmetric payoff matrices to allow asymmetric transactions between peers, and a learning-based [14] population dynamic model to specify the behavior of individual peers, which can be changed continuously. While social dilemmas have been studied extensively, P2P applications impose a unique set of challenges, including:

- **Large populations and high turnover:** A file sharing system such as Gnutella and KaZaa can exceed $100,000$ simultaneous users, and nodes can have an average life-time of the order of minutes [33].
- **Asymmetry of interest:** Asymmetric transactions of P2P systems create the possibility for asymmetry of interest. In the example in Figure 1, A wants service from B, B wants service from C, and C wants service from A.
- **Zero-cost identity:** Many P2P systems allow peers to continuously switch identities (i.e., *whitewash*).

Strategies that work well in traditional prisoners' dilemma games such as Tit-for-Tat [4] will not fare well in the P2P context. Therefore, we propose a family of scalable and robust incentive techniques, based upon a novel *Reciprocative* decision function, to address these challenges and provide different tradeoffs:

- **Discriminating Server Selection:** Cooperation requires familiarity between entities either directly or indirectly. However, the large populations and high turnover of P2P systems makes it less likely that repeat interactions will occur with a familiar entity. We show that by having each peer keep a

private history of the actions of other peers toward her, and using *discriminating* server selection, the Reciprocative decision function can scale to large populations and moderate levels of turnover.

- **Shared History:** Scaling to higher turnover and mitigating asymmetry of interest requires shared history. Consider the example in Figure 1. If everyone provides service, then the system operates optimally. However, if everyone keeps only private history, no one will provide service because B does not know that A has served C, etc. We show that with shared history, B knows that A served C and consequently will serve A. This results in a higher level of cooperation than with private history. The cost of shared history is a distributed infrastructure (e.g., distributed hash table-based storage) to store the history.

- **Maxflow-based Subjective Reputation:** Shared history creates the possibility for collusion. In the example in Figure 1, C can falsely claim that A served him, thus deceiving B into providing service. We show that a maxflow-based algorithm that computes reputation subjectively promotes cooperation despite collusion among 1/3 of the population. The basic idea is that B would only believe C if C had already provided service to B. The cost of the maxflow algorithm is its $O(V^3)$ running time, where V is the number of nodes in the system. To eliminate this cost, we have developed a constant mean running time variation, which trades effectiveness for complexity of computation. We show that the maxflow-based algorithm scales better than private history in the presence of colluders without the centralized trust required in previous work [9] [20].

- **Adaptive Stranger Policy:** Zero-cost identities allows non-cooperating peers to escape the consequences of not cooperating and eventually destroy cooperation in the system if not stopped. We show that if Reciprocative peers treat *strangers* (peers with no history) using a policy that adapts to the behavior of previous strangers, peers have little incentive to whitewash and whitewashing can be nearly eliminated from the system. The adaptive stranger policy does this without requiring centralized allocation of identities, an entry fee for newcomers, or rate-limiting [13] [9] [25].

- **Short-term History:** History also creates the possibility that a previously well-behaved peer with a good reputation will turn *traitor* and use his good reputation to exploit other peers. The peer could be making a strategic decision or someone may have hijacked her identity (e.g., by compromising her host). Long-term history exacerbates this problem by allowing peers with many previous transactions to exploit that history for many new transactions. We show that short-term history prevents traitors from disrupting cooperation.

The rest of the paper is organized as follows. We describe the model in Section 2 and the reciprocative decision function in Section 3. We then proceed to the incentive techniques in Section 4. In Section 4.1, we describe the challenges of large populations and high turnover and show the effectiveness of discriminating server selection and shared history. In Section 4.2, we describe collusion and demonstrate how subjective reputation mitigates it. In Section 4.3, we present the problem of zero-cost identities and show how an adaptive stranger policy promotes persistent identities. In Section 4.4, we show how traitors disrupt cooperation and how short-term history deals with them. We discuss related work in Section 5 and conclude in Section 6.

2. MODEL AND ASSUMPTIONS

In this section, we present our assumptions about P2P systems and their users, and introduce a model that aims to capture the behavior of users in a P2P system.

2.1 Assumptions

We assume a P2P system in which users are strategic, i.e., they act rationally to maximize their benefit. However, to capture some of the real-life unpredictability in the behavior of users, we allow users to randomly change their behavior with a low probability (see Section 2.4).

For simplicity, we assume a homogeneous system in which all peers issue and satisfy requests at the same rate. A peer can satisfy any request, and, unless otherwise specified, peers request service uniformly at random from the population.[1]. Finally, we assume that all transactions incur the same cost to all servers and provide the same benefit to all clients.

We assume that users can pollute shared history with false recommendations (Section 4.2), switch identities at zero-cost (Section 4.3), and spoof other users (Section 4.4). We do not assume any centralized trust or centralized infrastructure.

2.2 Model

To aid the development and study of the incentive schemes, in this section we present a model of the users' behaviors. In particular, we model the benefits and costs of P2P interactions (the *game*) and population dynamics caused by *mutation*, *learning*, and *turnover*. Our model is designed to have the following properties that characterize a large set of P2P systems:

- **Social Dilemma:** Universal cooperation should result in optimal overall utility, but individuals who exploit the cooperation of others while not cooperating themselves (i.e., *defecting*) should benefit more than users who do cooperate.

- **Asymmetric Transactions:** A peer may want service from another peer while not currently being able to provide the service that the second peer wants. Transactions should be able to have asymmetric payoffs.

- **Untraceable Defections:** A peer should not be able to determine the identity of peers who have defected on her. This models the difficulty or expense of determining that a peer could have provided a service, but didn't. For example, in the Gnutella file sharing system [21], a peer may simply ignore queries despite possessing the desired file, thus preventing the querying peer from identifying the defecting peer.

- **Dynamic Population:** Peers should be able to change their behavior and enter or leave the system independently and continuously.

[1]The exception is discussed in Section 4.1.1

		Server: Cooperate	Server: Defect
Client	Cooperate	R_c / R_s	S_c / T_s
	Defect	T_c / S_s	P_c / P_s

Figure 2: Payoff matrix for the Generalized Prisoner's Dilemma. T, R, P, and S stand for temptation, reward, punishment and sucker, respectively.

		Server: Provide Service	Server: Ignore Request
Client	Request Service	7 / -1	0 / 0
	Don't Request	0 / 0	0 / 0

Figure 3: The payoff matrix for an application like P2P file sharing or overlay routing.

2.3 Generalized Prisoner's Dilemma

The Prisoner's Dilemma, developed by Flood, Dresher, and Tucker in 1950 [22] is a non-cooperative repeated game satisfying the social dilemma requirement. Each game consists of two players who can defect or cooperate. Depending how each acts, the players receive a payoff. The players use a *strategy* to decide how to act. Unfortunately, existing work either uses a specific asymmetric payoff matrix or only gives the general form for a symmetric one [4].

Instead, we use the Generalized Prisoner's Dilemma (GPD), which specifies the general form for an asymmetric payoff matrix that preserves the social dilemma. In the GPD, one player is the client and one player is the server in each game, and it is only the decision of the server that is meaningful for determining the outome of the transaction. A player can be a client in one game and a server in another. The client and server receive the payoff from a generalized payoff matrix (Figure 2). R_c, S_c, T_c, and P_c are the client's payoff and R_s, S_s, T_s, and P_s are the server's payoff. A GPD payoff matrix must have the following properties to create a social dilemma:

1. Mutual cooperation leads to higher payoffs than mutual defection ($R_s + R_c > P_s + P_c$).
2. Mutual cooperation leads to higher payoffs than one player suckering the other ($R_s + R_c > S_c + T_s$ and $R_s + R_c > S_s + T_c$).
3. Defection dominates cooperation (at least weakly) at the individual level for the entity who decides whether to cooperate or defect: ($T_s \geq R_s$ and $P_s \geq S_s$ and ($T_s > R_s$ or $P_s > S_s$))

The last set of inequalities assume that clients do not incur a cost regardless of whether they cooperate or defect, and therefore clients always cooperate. These properties correspond to similar properties of the classic Prisoner's Dilemma and allow any form of asymmetric transaction while still creating a social dilemma.

Furthermore, one or more of the four possible actions (client cooperate and defect, and server cooperate and defect) can be untraceable. If one player makes an untraceable action, the other player does not know the identity of the first player.

For example, to model a P2P application like file sharing or overlay routing, we use the specific payoff matrix values shown in Figure 3. This satisfies the inequalities specified above, where only the server can choose between cooperating and defecting. In addition, for this particular payoff matrix, clients are unable to trace server defections. This is the payoff matrix that we use in our simulation results.

2.4 Population Dynamics

A characteristic of P2P systems is that peers change their behavior and enter or leave the system independently and continuously. Several studies [4] [28] of repeated Prisoner's Dilemma games use an evolutionary model [19] [34] of population dynamics. An evolutionary model is not suitable for P2P systems because it only specifies the global behavior and all changes occur at discrete times. For example, it may specify that a population of 5 "100% Cooperate" players and 5 "100% Defect" players evolves into a population with 3 and 7 players, respectively. It does not specify which specific players switched. Furthermore, all the switching occurs at the end of a generation instead of continuously, like in a real P2P system. As a result, evolutionary population dynamics do not accurately model turnover, traitors, and strangers.

In our model, entities take independent and continuous actions that change the composition of the population. Time consists of *rounds*. In each round, every player plays one game as a client and one game as a server. At the end of a round, a player may: 1) *mutate* 2) *learn*, 3) *turnover*, or 4) stay the same. If a player mutates, she switches to a randomly picked strategy. If she learns, she switches to a strategy that she believes will produce a higher score (described in more detail below). If she maintains her identity after switching strategies, then she is referred to as a traitor. If a player suffers turnover, she leaves the system and is replaced with a newcomer who uses the same strategy as the exiting player.

To learn, a player collects local information about the performance of different strategies. This information consists of both her personal observations of strategy performance and the observations of those players she interacts with. This models users communicating out-of-band about how strategies perform. Let s be the running average of the performance of a player's current strategy per round and age be the number of rounds she has been using the strategy. A strategy's *rating* is

$$\frac{RunningAverage(s * age)}{RunningAverage(age)}.$$

We use the age and compute the running average before the ratio to prevent young samples (which are more likely to be outliers) from skewing the rating. At the end of a round, a player switches to highest rated strategy with a probability proportional to the difference in score between her current strategy and the highest rated strategy.

3. RECIPROCATIVE DECISION FUNCTION

In this section, we present the new decision function, *Reciprocative*, that is the basis for our incentive techniques. A *decision function* maps from a history of a player's actions to a decision whether to cooperate with or defect on that player. A strategy consists of a decision function, private or shared history, a server selection mechanism, and a stranger policy. Our approach to incentives is to design strategies which maximize both individual and social benefit. Strategic users will choose to use such strategies and thereby drive the system to high levels of cooperation. Two examples of simple decision functions are "100% Cooperate" and "100% Defect". "100% Cooperate" models a naive user who does not yet realize that she is being exploited. "100% Defect" models a greedy user who is intent on exploiting the system. In the absence of incentive techniques, "100% Defect" users will quickly dominate the "100% Cooperate" users and destroy cooperation in the system.

Our requirements for a decision function are that (1) it can use shared and subjective history, (2) it can deal with untraceable defections, and (3) it is robust against different patterns of defection. Previous decision functions such as Tit-for-Tat[4] and Image[28] (see Section 5) do not satisfy these criteria. For example, Tit-for-Tat and Image base their decisions on both cooperations and defections, therefore cannot deal with untraceable defections . In this section and the remaining sections we demonstrate how the Reciprocative-based strategies satisfy all of the requirements stated above.

The probability that a Reciprocative player cooperates with a peer is a function of its *normalized generosity*. Generosity measures the benefit an entity has provided relative to the benefit it has consumed. This is important because entities which consume more services than they provide, even if they provide many services, will cause cooperation to collapse. For some entity i, let p_i and c_i be the services i has provided and consumed, respectively. Entity i's generosity is simply the ratio of the service it provides to the service it consumes:

$$g(i) = p_i/c_i. \quad (1)$$

One possibility is to cooperate with a probability equal to the generosity. Although this is effective in some cases, in other cases, a Reciprocative player may consume more than she provides (e.g., when initially using the "Stranger Defect" policy in 4.3). This will cause Reciprocative players to defect on each other. To prevent this situation, a Reciprocative player uses its own generosity as a measuring stick to judge its peer's generosity. *Normalized* generosity measures entity i's generosity relative to entity j's generosity. More concretely, entity i's *normalized* generosity as perceived by entity j is

$$g_j(i) = g(i)/g(j). \quad (2)$$

In the remainder of this section, we describe our simulation framework, and use it to demonstrate the benefits of the baseline Reciprocative decision function.

Parameter	Nominal value	Section
Population Size	100	2.4
Run Time	1000 rounds	2.4
Payoff Matrix	File Sharing	2.3
Ratio using "100% Cooperate"	1/3	3
Ratio using "100% Defect"	1/3	3
Ratio using Reciprocative	1/3	3
Mutation Probability	0.0	2.4
Learning Probability	0.05	2.4
Turnover Probability	0.0001	2.4
Hit Rate	1.0	4.1.1

Table 1: Default simulation parameters.

3.1 Simulation Framework

Our simulator implements the model described in Section 2. We use the asymmetric file sharing payoff matrix (Figure 3) with untraceable defections because it models transactions in many P2P systems like file-sharing and packet forwarding in ad hoc and overlay networks. Our simulation study is composed of different scenarios reflecting the challenges of various non-cooperative behaviors. Table 1 presents the nominal parameter values used in our simulation. The "Ratio using" rows refer to the initial ratio of the total population using a particular strategy. In each scenario we vary the value range of a specific parameter to reflect a particular situation or attack. We then vary the exact properties of the Reciprocative strategy to defend against that situation or attack.

3.2 Baseline Results

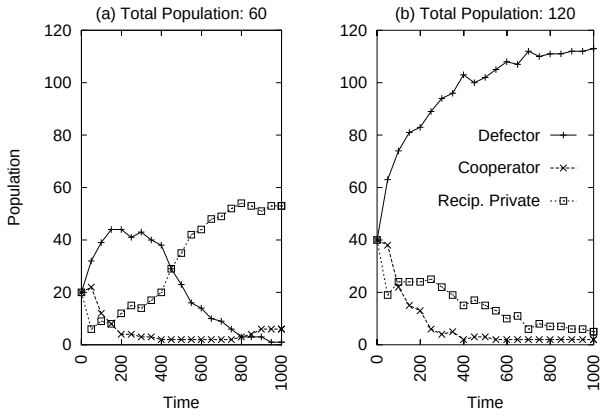

Figure 4: The evolution of strategy populations over time. "Time" the number of elapsed rounds. "Population" is the number of players using a strategy.

In this section, we present the dynamics of the game for the basic scenario presented in Table 1 to familiarize the reader and set a baseline for more complicated scenarios. Figures 4(a) (60 players) and (b) (120 players) show players switching to higher scoring strategies over time in two separate runs of the simulator. Each point in the graph represents the number of players using a particular strategy at one point in time. Figures 5(a) and (b) show the corresponding mean overall score per round. This measures the degree of cooperation in the system: 6 is the maximum possible (achieved when everybody cooperates) and 0 is the minimum (achieved when everybody defects). From the file sharing payoff matrix, a net of 6 means everyone is able to download a file and a 0 means that no one

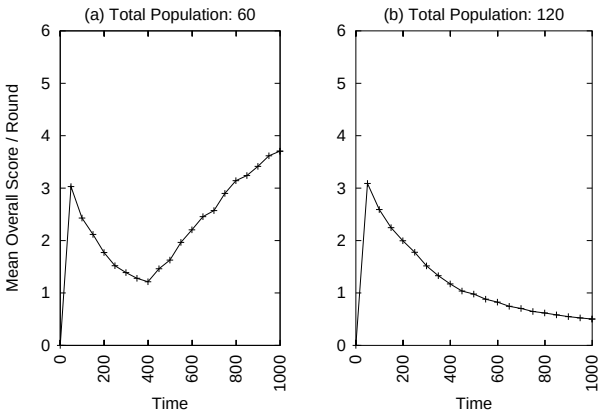

Figure 5: The mean overall per round score over time.

is able to do so. We use this metric in all later results to evaluate our incentive techniques.

Figure 5(a) shows that the Reciprocative strategy using private history causes a system of 60 players to converge to a cooperation level of 3.7, but drops to 0.5 for 120 players. One would expect the 60 player system to reach the optimal level of cooperation (6) because all the defectors are eliminated from the system. It does not because of asymmetry of interest. For example, suppose player B is using Reciprocative with private history. Player A may happen to ask for service from player B twice in succession without providing service to player B in the interim. Player B does not know of the service player A has provided to others, so player B will reject service to player A, even though player A is cooperative. We discuss solutions to asymmetry of interest and the failure of Reciprocative in the 120 player system in Section 4.1.

4. RECIPROCATIVE-BASED INCENTIVE TECHNIQUES

In this section we present our incentives techniques and evaluate their behavior by simulation. To make the exposition clear we group our techniques by the challenges they address: large populations and high turnover (Section 4.1), collusions (Section 4.2), zero-cost identities (Section 4.3), and traitors (Section 4.4).

4.1 Large Populations and High Turnover

The large populations and high turnover of P2P systems makes it less likely that repeat interactions will occur with a familiar entity. Under these conditions, basing decisions only on private history (records about interactions the peer has been directly involved in) is not effective. In addition, private history does not deal well with asymmetry of interest. For example, if player B has cooperated with others but not with player A himself in the past, player A has no indication of player B's generosity, thus may unduly defect on him. We propose two mechanisms to alleviate the problem of few repeat transactions: server-selection and shared history.

4.1.1 Server Selection

A natural way to increase the probability of interacting with familiar peers is by discriminating server selection. However, the asymmetry of transactions challenges selection mechanisms. Unlike in the prisoner's dilemma payoff matrix, where players can benefit one another within a single transaction, transactions in GPD are asymmetric. As a result, a player who selects her donor for the second time without contributing to her in the interim may face a defection. In addition, due to untraceability of defections, it is impossible to maintain blacklists to avoid interactions with known defectors.

In order to deal with asymmetric transactions, every player holds (fixed size) lists of both past donors and past recipients, and selects a server from one of these lists at random with equal probabilities. This way, users approach their past recipients and give them a chance to reciprocate.

In scenarios with selective users we omit the complete availability assumption to prevent players from being clustered into a lot of very small groups; thus, we assume that every player can perform the requested service with probability p (for the results presented in this section, $p = .3$). In addition, in order to avoid bias in favor of the selective players, *all* players (including the non-discriminative ones) select servers for games.

Figure 6 demonstrates the effectiveness of the proposed selection mechanism in scenarios with large population sizes. We fix the initial ratio of Reciprocative in the population (33%) while varying the population size (between 24 to 1000) (Notice that while in Figures 4(a) and (b), the data points demonstrates the evolution of the system over time, each data point in this figure is the result of an entire simulation for a specific scenario). The figure shows that the Reciprocative decision function using private history in conjunction with selective behavior can scale to large populations.

In Figure 7 we fix the population size and vary the turnover rate. It demonstrates that while selective behavior is effective for low turnover rates, as turnover gets higher, selective behavior does not scale. This occurs because selection is only effective as long as players from the past stay alive for long enough such that they can be selected for future games.

4.1.2 Shared history

In order to mitigate asymmetry of interest and scale to higher turnover rate, there is a need in shared history. Shared history means that every peer keeps records about all of the interactions that occur in the system, regardless of whether he was directly involved in them or not. It allows players to leverage off of the experiences of others in cases of few repeat transactions. It only requires that someone has interacted with a particular player for the entire population to observe it, thus scales better to large populations and high turnovers, and also tolerates asymmetry of interest. Some examples of shared history schemes are [20] [23] [28].

Figure 7 shows the effectiveness of shared history under high turnover rates. In this figure, we fix the population size and vary the turnover rate. While selective players with private history can only tolerate a moderate turnover, shared history scales to turnovers of up to approximately 0.1. This means that 10% of the players leave the system at the end of each round. In Figure 6 we fix the turnover and vary the population size. It shows that shared history causes the system to converge to optimal cooperation and performance, regardless of the size of the population.

These results show that shared history addresses all three challenges of large populations, high turnover, and asymmetry of transactions. Nevertheless, shared history has two disadvantages. First,

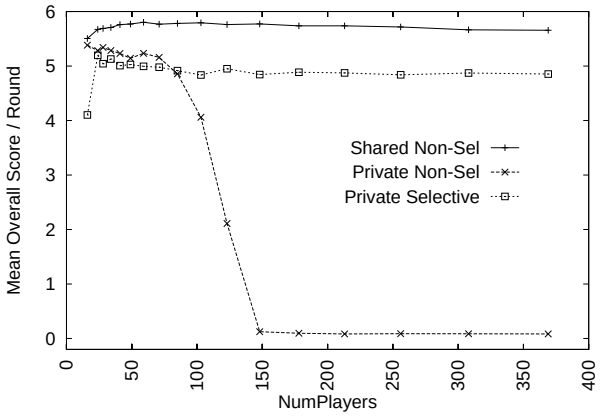

Figure 6: Private vs. Shared History as a function of population size.

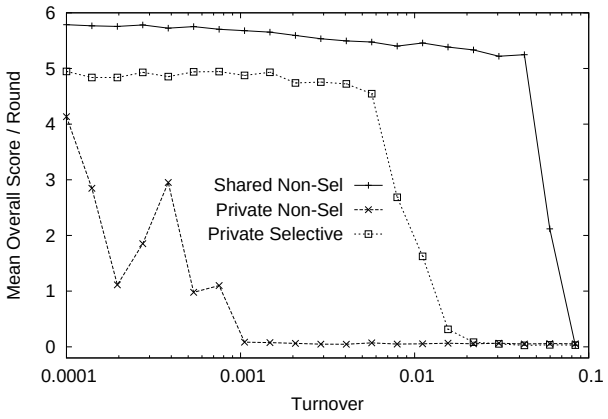

Figure 7: Performance of selection mechanism under turnover. The x-axis is the turnover rate. The y-axis is the mean overall per round score.

while a decentralized implementation of private history is straightforward, implementation of shared-history requires communication overhead or centralization. A decentralized shared history can be implemented, for example, on top of a DHT, using a peer-to-peer storage system [36] or by disseminating information to other entities in a similar way to routing protocols. Second, and more fundamental, shared history is vulnerable to collusion. In the next section we propose a mechanism that addresses this problem.

4.2 Collusion and Other Shared History Attacks

4.2.1 Collusion

While shared history is scalable, it is vulnerable to collusion. Collusion can be either positive (e.g. defecting entities claim that other defecting entities cooperated with them) or negative (e.g. entities claim that other cooperative entities defected on them). Collusion subverts any strategy in which everyone in the system agrees on the reputation of a player (*objective reputation*). An example of objective reputation is to use the Reciprocative decision function with shared history to count the total number of cooperations a player has given to and received from all entities in the system; another example is the Image strategy [28]. The effect of collusion is magnified in systems with zero-cost identities, where users can create fake identities that report false statements.

Instead, to deal with collusion, entities can compute reputation *subjectively*, where player A weighs player B's opinions based on how much player A trusts player B. Our subjective algorithm is based on maxflow [24] [32]. Maxflow is a graph theoretic problem, which given a directed graph with weighted edges asks what is the greatest rate at which "material" can be shipped from the source to the target without violating any capacity constraints. For example, in figure 8 each edge is labeled with the amount of traffic that can travel on it. The maxflow algorithm computes the maximum amount of traffic that can go from the source (s) to the target (t) without violating the constraints. In this example, even though there is a loop of high capacity edges, the maxflow between the source and the target is only 2 (the numbers in brackets represent the actual flow on each edge in the solution).

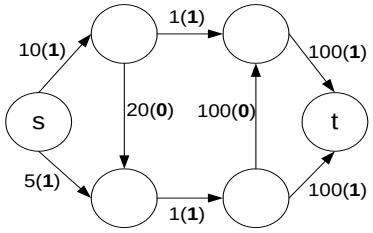

Figure 8: Each edge in the graph is labeled with its capacity and the actual flow it carries in brackets. The maxflow between the source and the target in the graph is 2.

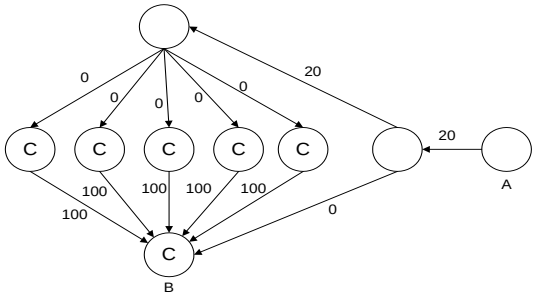

Figure 9: This graph illustrates the robustness of maxflow in the presence of colluders who report bogus high reputation values.

We apply the maxflow algorithm by constructing a graph whose vertices are entities and the edges are the services that entities have received from each other. This information can be stored using the same methods as the shared history. A maxflow is the greatest level of reputation the source can give to the sink without violating "reputation capacity" constraints. As a result, nodes who dishonestly report high reputation values will not be able to subvert the reputation system.

Figure 9 illustrates a scenario in which all the colluders (labeled with C) report high reputation values for each other. When node A computes the subjective reputation of B using the maxflow algorithm, it will not be affected by the local false reputation values, rather the maxflow in this case will be 0. This is because no service has been received from any of the colluders.

In our algorithm, the benefit that entity i has received (indirectly) from entity j is the maxflow from j to i. Conversely, the benefit that entity i has provided indirectly to j is the maxflow from i to j. The subjective reputation of entity j as perceived by i is:

$$min\left(\frac{maxflow(j\ to\ i)}{maxflow(i\ to\ j)}, 1\right) \quad (3)$$

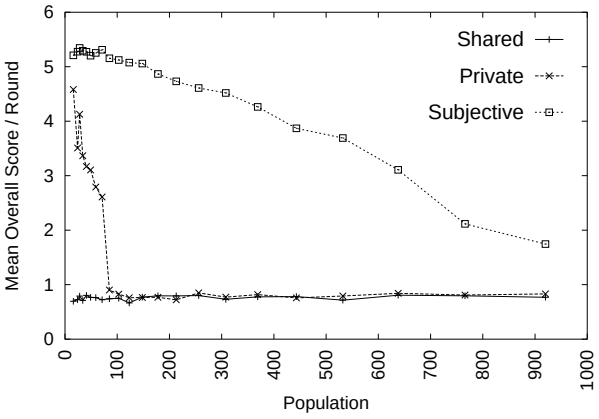

Figure 10: Subjective shared history compared to objective shared history and private history in the presence of colluders.

Algorithm 1 CONSTANTTIMEMAXFLOW Bound the mean running time of Maxfbw to a constant.

method $CTMaxflow(self, src, dst)$

1: $self.surplus \leftarrow self.surplus + self.increment$
{Use the running mean as a prediction.}
2: **if** $random() > (0.5 * self.surplus / self.mean\_iterations)$ **then**
3: return $None$ {Not enough surplus to run.}
4: **end if**
{Get the fbw and number of iterations used from the maxfbw alg.}
5: $flow, iterations \leftarrow Maxflow(self.G, src, dst)$
6: $self.surplus \leftarrow self.surplus - iterations$
{Keep a running mean of the number of iterations used.}
7: $self.mean\_iterations \leftarrow self.\alpha * self.mean\_iterations + (1 - self.\alpha) * iterations$
8: return $flow$

The cost of maxflow is its long running time. The standard preflow-push maxflow algorithm has a worst case running time of $O(V^3)$. Instead, we use Algorithm 1 which has a constant mean running time, but sometimes returns no flow even though one exists. The essential idea is to bound the mean number of nodes examined during the maxflow computation. This bounds the overhead, but also bounds the effectiveness. Despite this, the results below show that a maxflow-based Reciprocative decision function scales to higher populations than one using private history.

Figure 10 compares the effectiveness of subjective reputation to objective reputation in the presence of colluders. In these scenarios, defectors collude by claiming that other colluders that they encounter gave them 100 cooperations for that encounter. Also, the parameters for Algorithm 1 are set as follows: $increment = 100$, $\alpha = 0.9$.

As in previous sections, Reciprocative with private history results in cooperation up to a point, beyond which it fails. The difference here is that objective shared history fails for all population sizes. This is because the Reciprocative players cooperate with the colluders because of their high reputations. However, subjective history can reach high levels of cooperation regardless of colluders. This is because there are no high weight paths in the cooperation graph from colluders to any non-colluders, so the maxflow from a colluder to any non-colluder is 0. Therefore, a subjective Reciprocative player will conclude that that colluder has not provided any service to her and will reject service to the colluder. Thus, the maxflow algorithm enables Reciprocative to maintain the scalability of shared history without being vulnerable to collusion or requiring centralized trust (e.g., trusted peers). Since we bound the running time of the maxflow algorithm, cooperation decreases as the population size increases, but the key point is that the subjective Reciprocative decision function scales to higher populations than one using private history. This advantage only increases over time as CPU power increases and more cycles can be devoted to running the maxflow algorithm (by increasing the $increment$ parameter).

Despite the robustness of the maxflow algorithm to the simple form of collusion described previously, it still has vulnerabilities to more sophisticated attacks. One is for an entity (the "mole") to provide service and then lie positively about other colluders. The other colluders can then exploit their reputation to receive service. However, the effectiveness of this attack relies on the amount of service that the mole provides. Since the mole is paying all of the cost of providing service and receiving none of the benefit, she has a strong incentive to stop colluding and try another strategy. This forces the colluders to use mechanisms to maintain cooperation within their group, which may drive the cost of collusion to exceed the benefit.

4.2.2 False reports

Another attack is for a defector to lie about receiving or providing service to another entity. There are four possibile actions that can be lied about: providing service, not providing service, receiving service, and not receiving service. Falsely claiming to receive service is the simple collusion attack described above. Falsely claiming not to have provided service provides no benefit to the attacker.

Falsely claiming to have provided service or not to have received it allows an attacker to boost her own reputation and/or lower the reputation of another entity. An entity may want to lower another entity's reputation in order to discourage others from selecting it and exclusively use its service. These false claims are clearly identifiable in the shared history as inconsistencies where one entity claims a transaction occurred and another claims it did not. To limit this attack, we modify the maxflow algorithm so that an entity always believes the entity that is closer to him in the flow graph. If both entities are equally distant, then the disputed edge in the flow is not critical to the evaluation and is ignored. This modification prevents those cases where the attacker is making false claims about an entity that is closer than her to the evaluating entity, which prevents her from boosting her own reputation. The remaining possibilities are for the attacker to falsely claim to have provided service to or not to have received it from a victim entity that is farther from the evalulator than her. In these cases, an attacker can only lower the reputation of the victim. The effectiveness of doing this is limited by the number of services provided and received by the attacker, which makes executing this attack expensive.

4.3 Zero-Cost Identities

History assumes that entities maintain persistent identities. However, in most P2P systems, identities are *zero-cost*. This is desirable for network growth as it encourages newcomers to join the system. However, this also allows misbehaving users to escape the consequences of their actions by switching to new identities (i.e., *whitewashing*). Whitewashers can cause the system to collapse if they are not punished appropriately. Unfortunately, a player cannot tell if a stranger is a whitewasher or a legitimate *newcomer*. Always cooperating with strangers encourages newcomers to join, but at the same time encourages whitewashing behavior. Always defecting on strangers prevents whitewashing, but discourages newcomers from joining and may also initiate unfavorable cycles of defection.

This tension suggests that any stranger policy that has a fixed probability of cooperating with strangers will fail by either being too stingy when most strangers are newcomers or too generous when most strangers are whitewashers. Our solution is the "Stranger Adaptive" stranger policy. The idea is to be generous to strangers when they are being generous and stingy when they are stingy.

Let p_s and c_s be the number of services that strangers have provided and consumed, respectively. The probability that a player using "Stranger Adaptive" helps a stranger is p_s/c_s. However, we do not wish to keep these counts permanently (for reasons described in Section 4.4). Also, players may not know when strangers defect because defections are untraceable (as described in Section 2). Consequently, instead of keeping p_s and c_s, we assume that $k = p_s + c_s$, where k is a constant and we keep the running ratio $r = p_s/c_s$. When we need to increment p_s or c_s, we generate the current values of p_s and c_s from k and r:

$$c_s = k/(1+r)$$

$$p_s = c_s * r$$

We then compute the new r as follows:

$$r = (p_s + 1)/c_s \text{ , if the stranger provided service}$$

$$r = p_s/(c_s + 1) \text{ , if the stranger consumed service}$$

This method allows us to keep a running ratio that reflects the recent generosity of strangers without knowing when strangers have defected.

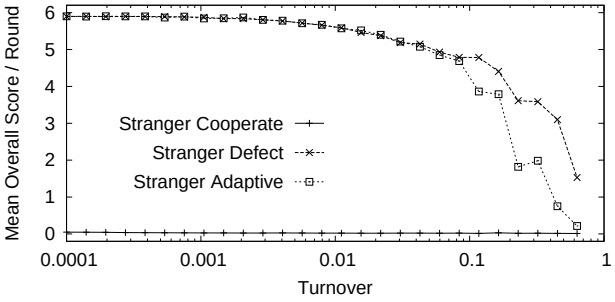

Figure 11: Different stranger policies for Reciprocative with shared history. The x-axis is the turnover rate on a log scale. The y-axis is the mean overall per round score.

Figures 11 and 12 compare the effectiveness of the Reciprocative strategy using different policies toward strangers. Figure 11 compares different stranger policies for Reciprocative with shared history, while Figure 12 is with private history. In both figures, the players using the "100% Defect" strategy change their identity (whitewash) after every transaction and are indistinguishable from legitimate newcomers. The Reciprocative players using the "Stranger Cooperate" policy completely fail to achieve cooperation. This stranger policy allows whitewashers to maximize their payoff and consequently provides a high incentive for users to switch to whitewashing.

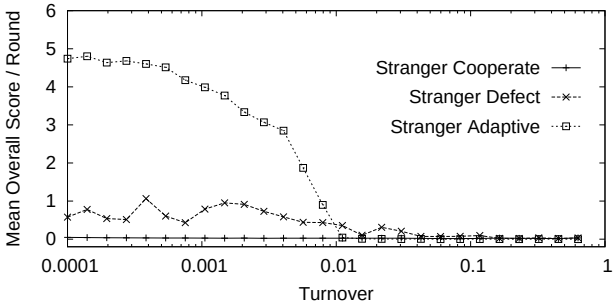

Figure 12: Different stranger policies for Reciprocative with private history. The x-axis is the turnover rate on a log scale. The y-axis is the mean overall per round score.

In contrast, Figure 11 shows that the "Stranger Defect" policy is effective with shared history. This is because whitewashers always appear to be strangers and therefore the Reciprocative players will always defect on them. This is consistent with previous work [13] showing that punishing strangers deals with whitewashers. However, Figure 12 shows that "Stranger Defect" is not effective with private history. This is because Reciprocative requires some initial cooperation to bootstrap. In the shared history case, a Reciprocative player can observe that another player has already cooperated with others. With private history, the Reciprocative player only knows about the other players' actions toward her. Therefore, the initial defection dictated by the "Stranger Defect" policy will lead to later defections, which will prevent Reciprocative players from ever cooperating with each other. In other simulations not shown here, the "Stranger Defect" stranger policy fails even with shared history when there are no initial "100% Cooperate" players.

Figure 11 shows that with shared history, the "Stranger Adaptive" policy performs as well as "Stranger Defect" policy until the turnover rate is very high (10% of the population turning over after every transaction). In these scenarios, "Stranger Adaptive" is using $k = 10$ and each player keeps a private r. More importantly, it is significantly better than "Stranger Defect" policy with private history because it can bootstrap cooperation. Although the "Stranger Defect" policy is marginally more effective than "Stranger Adaptive" at very high rates of turnover, P2P systems are unlikely to operate there because other services (e.g., routing) also cannot tolerate very high turnover.

We conclude that of the stranger policies that we have explored, "Stranger Adaptive" is the most effective. By using "Stranger Adaptive", P2P systems with zero-cost identities and a sufficiently low turnover can sustain cooperation without a centralized allocation of identities.

4.4 Traitors

Traitors are players who acquire high reputation scores by cooperating for a while, and then traitorously turn into defectors before leaving the system. They model both users who turn deliberately to gain a higher score and cooperators whose identities have been stolen and exploited by defectors. A strategy that maintains long-term history without discriminating between old and recent actions becomes highly vulnerable to exploitation by these traitors.

The top two graphs in Figure 13 demonstrate the effect of traitors on cooperation in a system where players keep long-term history (never clear history). In these simulations, we run for 2000 rounds and allow cooperative players to keep their identities when switching to the 100% Defector strategy. We use the default values for the other parameters. Without traitors, the cooperative strategies thrive. With traitors, the cooperative strategies thrive until a cooperator turns traitor after 600 rounds. As this "cooperator" exploits her reputation to achieve a high score, other cooperative players notice this and follow suit via learning. Cooperation eventually collapses. On the other hand, if we maintain short-term history and/or discounting ancient history vis-a-vis recent history, traitors can be quickly detected, and the overall cooperation level stays high, as shown in the bottom two graphs in Figure 13.

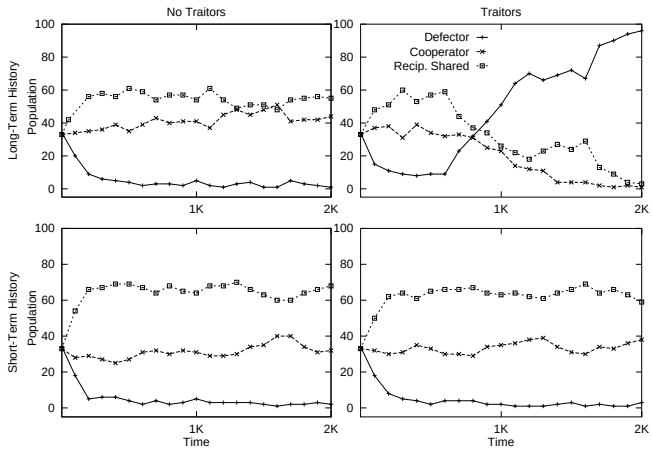

Figure 13: Keeping long-term vs. short-term history both with and without traitors.

5. RELATED WORK

Previous work has examined the incentive problem as applied to societies in general and more recently to Internet applications and peer-to-peer systems in particular. A well-known phenomenon in this context is the "tragedy of the commons" [18] where resources are under-provisioned due to selfish users who free-ride on the system's resources, and is especially common in large networks [29] [3].

The problem has been extensively studied adopting a game theoretic approach. The prisoners' dilemma model provides a natural framework to study the effectiveness of different strategies in establishing cooperation among players. In a simulation environment with many repeated games, persistent identities, and no collusion, Axelrod [4] shows that the Tit-for-Tat strategy dominates. Our model assumes growth follows local learning rather than evolutionary dynamics [14], and also allows for more kinds of attacks. Nowak and Sigmund [28] introduce the Image strategy and demonstrate its ability to establish cooperation among players despite few repeat transactions by the employment of shared history. Players using Image cooperate with players whose global count of cooperations minus defections exceeds some threshold. As a result, an Image player is either vulnerable to partial defectors (if the threshold is set too low) or does not cooperate with other Image players (if the threshold is set too high).

In recent years, researchers have used economic "mechanism design" theory to tackle the cooperation problem in Internet applications. Mechanism design is the inverse of game theory. It asks how to design a game in which the behavior of strategic players results in the socially desired outcome. Distributed Algorithmic Mechanism Design seeks solutions within this framework that are both fully distributed and computationally tractable [12]. [10] and [11] are examples of applying DAMD to BGP routing and multicast cost sharing. More recently, DAMD has been also studied in dynamic environments [38]. In this context, demonstrating the superiority of a cooperative strategy (as in the case of our work) is consistent with the objective of incentivizing the desired behavior among selfish players.

The unique challenges imposed by peer-to-peer systems inspired additional body of work [5] [37], mainly in the context of packet forwarding in wireless ad-hoc routing [8] [27] [30] [35], and file sharing [15] [31]. Friedman and Resnick [13] consider the problem of zero-cost identities in online environments and find that in such systems punishing all newcomers is inevitable. Using a theoretical model, they demonstrate that such a system can converge to cooperation only for sufficiently low turnover rates, which our results confirm. [6] and [9] show that whitewashing and collusion can have dire consequences for peer-to-peer systems and are difficult to prevent in a fully decentralized system.

Some commercial file sharing clients [1] [2] provide incentive mechanisms which are enforced by making it difficult for the user to modify the source code. These mechanisms can be circumvented by a skilled user or by a competing company releasing a compatible client without the incentive restrictions. Also, these mechanisms are still vulnerable to zero-cost identities and collusion. BitTorrent [7] uses Tit-for-Tat as a method for resource allocation, where a user's upload rate dictates his download rate.

6. CONCLUSIONS

In this paper we take a game theoretic approach to the problem of cooperation in peer-to-peer networks. Addressing the challenges imposed by P2P systems, including large populations, high turnover, asymmetry of interest and zero-cost identities, we propose a family of scalable and robust incentive techniques, based upon the Reciprocative decision function, to support cooperative behavior and improve overall system performance.

We find that the adoption of shared history and discriminating server selection techniques can mitigate the challenge of few repeat transactions that arises due to large population size, high turnover and asymmetry of interest. Furthermore, cooperation can be established even in the presence of zero-cost identities through the use of an adaptive policy towards strangers. Finally, colluders and traitors can be kept in check via subjective reputations and short-term history, respectively.

7. ACKNOWLEDGMENTS

We thank Mary Baker, T.J. Giuli, Petros Maniatis, the anonymous reviewer, and our shepherd, Margo Seltzer, for their useful comments that helped improve the paper. This work is supported in part by the National Science Foundation under ITR awards ANI-0085879 and ANI-0331659, and Career award ANI-0133811. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF, or the U.S. government.

8. REFERENCES

[1] Kazaa. http://www.kazaa.com.

[2] Limewire. http://www.limewire.com.

[3] ADAR, E., AND HUBERMAN, B. A. Free Riding on Gnutella. *First Monday 5*, 10 (October 2000).

[4] AXELROD, R. *The Evolution of Cooperation*. Basic Books, 1984.

[5] BURAGOHAIN, C., AGRAWAL, D., AND SURI, S. A Game-Theoretic Framework for Incentives in P2P Systems. In *International Conference on Peer-to-Peer Computing* (Sep 2003).

[6] CASTRO, M., DRUSCHEL, P., GANESH, A., ROWSTRON, A., AND WALLACH, D. S. Security for Structured Peer-to-Peer Overlay Networks. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)* (2002).

[7] COHEN, B. Incentives build robustness in bittorrent. In *1st Workshop on Economics of Peer-to-Peer Systems* (2003).

[8] CROWCROFT, J., GIBBENS, R., KELLY, F., AND ˘ OSTRING, S. Modeling Incentives for Collaboration in Mobile Ad Hoc Networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks* (2003).

[9] DOUCEUR, J. R. The Sybil Attack. In *Electronic Proceedings of the International Workshop on Peer-to-Peer Systems* (2002).

[10] FEIGENBAUM, J., PAPADIMITRIOU, C., SAMI, R., AND SHENKER, S. A BGP-based Mechanism for Lowest-Cost Routing. In *Proceedings of the ACM Symposium on Principles of Distributed Computing* (2002).

[11] FEIGENBAUM, J., PAPADIMITRIOU, C., AND SHENKER, S. Sharing the Cost of Multicast Transmissions. In *Journal of Computer and System Sciences* (2001), vol. 63, pp. 21–41.

[12] FEIGENBAUM, J., AND SHENKER, S. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications* (2002).

[13] FRIEDMAN, E., AND RESNICK, P. The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy 10*, 2 (1998), 173–199.

[14] FUDENBERG, D., AND LEVINE, D. K. *The Theory of Learning in Games*. The MIT Press, 1999.

[15] GOLLE, P., LEYTON-BROWN, K., MIRONOV, I., AND LILLIBRIDGE, M. Incentives For Sharing in Peer-to-Peer Networks. In *Proceedings of the 3rd ACM conference on Electronic Commerce, October 2001* (2001).

[16] GROSS, B., AND ACQUISTI, A. Balances of Power on EBay: Peers or Unquals? In *Workshop on economics of peer-to-peer networks* (2003).

[17] GU, B., AND JARVENPAA, S. Are Contributions to P2P Technical Forums Private or Public Goods? – An Empirical Investigation. In *1st Workshop on Economics of Peer-to-Peer Systems* (2003).

[18] HARDIN, G. The Tragedy of the Commons. *Science 162* (1968), 1243–1248.

[19] JOSEF HOFBAUER AND KARL SIGMUND. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

[20] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the Twelfth International World Wide Web Conference* (May 2003).

[21] KAN, G. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, 1st ed. O'Reilly & Associates, Inc., March 2001, ch. Gnutella, pp. 94–122.

[22] KUHN, S. Prisoner's Dilemma. In *The Stanford Encyclopedia of Philosophy*, Edward N. Zalta, Ed., Summer ed. 2003.

[23] LEE, S., SHERWOOD, R., AND BHATTACHARJEE, B. Cooperative Peer Groups in Nice. In *Proceedings of the IEEE INFOCOM* (2003).

[24] LEVIEN, R., AND AIKEN, A. Attack-Resistant Trust Metrics for Public Key Certifi cation. In *Proceedings of the USENIX Security Symposium* (1998), pp. 229–242.

[25] MANIATIS, P., ROUSSOPOULOS, M., GIULI, T. J., ROSENTHAL, D. S. H., BAKER, M., AND MULIADI, Y. Preserving Peer Replicas by Rate-Limited Sampled Voting. In *ACM Symposium on Operating Systems Principles* (2003).

[26] MARTI, S., GIULI, T. J., LAI, K., AND BAKER, M. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of MobiCom* (2000), pp. 255–265.

[27] MICHIARDI, P., AND MOLVA, R. A Game Theoretical Approach to Evaluate Cooperation Enforcement Mechanisms in Mobile Ad Hoc Networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks* (2003).

[28] NOWAK, M. A., AND SIGMUND, K. Evolution of Indirect Reciprocity by Image Scoring. *Nature 393* (1998), 573–577.

[29] OLSON, M. *The Logic of Collective Action: Public Goods and the Theory of Groups*. Harvard University Press, 1971.

[30] RAGHAVAN, B., AND SNOEREN, A. Priority Forwarding in Ad Hoc Networks with Self-Ineterested Parties. In *Workshop on Economics of Peer-to-Peer Systems* (June 2003).

[31] RANGANATHAN, K., RIPEANU, M., SARIN, A., AND FOSTER, I. To Share or Not to Share: An Analysis of Incentives to Contribute in Collaborative File Sharing Environments. In *Workshop on Economics of Peer-to-Peer Systems* (June 2003).

[32] REITER, M. K., AND STUBBLEBINE, S. G. Authentication Metric Analysis and Design. *ACM Transactions on Information and System Security 2*, 2 (1999), 138–158.

[33] SAROIU, S., GUMMADI, P. K., AND GRIBBLE, S. D. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)* (2002).

[34] SMITH, J. M. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[35] URPI, A., BONUCCELLI, M., AND GIORDANO, S. Modeling Cooperation in Mobile Ad Hoc Networks: a Formal Description of Selfi shness. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks* (2003).

[36] VISHNUMURTHY, V., CHANDRAKUMAR, S., AND SIRER, E. G. KARMA : A Secure Economic Framework for P2P Resource Sharing. In *Workshop on Economics of Peer-to-Peer Networks* (2003).

[37] WANG, W., AND LI, B. To Play or to Control: A Game-based Control-Theoretic Approach to Peer-to-Peer Incentive Engineering. In *International Workshop on Quality of Service* (June 2003).

[38] WOODARD, C. J., AND PARKES, D. C. Strategyproof mechanisms for ad hoc network formation. In *Workshop on Economics of Peer-to-Peer Systems* (June 2003).