



## Continuously Rounding in Peer-to-Peer Networks

Nennuru Riyaz Ali

Department of CSE

Annamacharya Institute of Technology and Sciences  
India

K.Chandra Babu

Assistant professor, Department of CSE

Annamacharya Institute of Technology and Sciences  
India

**Abstract**— We quantify the effectiveness of random walks for searching and construction of unstructured peer-to-peer (P2P) networks. For searching, we argue that random walks achieve improvement over loading in the case of clustered overlay topologies and in the case of re-issuing the same request several times. For construction, we argue that an expander can be maintained dynamically with constant operations per addition. The key technical ingredient of our approach is a deep result of stochastic processes indicating that samples taken from consecutive steps of a random walk can achieve statistical properties similar to independent sampling.

**Keywords**— Graph theory, Combinatory, Statistics, peer-to-peer networks.

### I. INTRODUCTION

Many peer-to-peer (P2P) systems rely on cooperation among self-interested users. For example, users of file-sharing systems who do not share their own resources cause long delays or download failures. When non-cooperative users benefit from free-riding on others' resources, the "tragedy of the commons" [10] is inevitable. Avoiding this problem requires incentives for cooperation. To model this problem, we use the Evolutionary Prisoner's Dilemma (EPD) [3] to capture the tension between individual and social utility. In the situations that EPD characterizes, cooperation requires repetition and reputation. Both techniques increase familiarity between entities (either directly or indirectly), thereby reducing the probability of interactions with strangers and consequently enhancing cooperation. The context of P2P applications, however, imposes new challenges. First, the large scale of these systems makes it less likely that repeat interactions will occur with the same entity. Second, reputation assumes that players maintain persistent identities, but the existence of zero-cost identities in many P2P systems allows entities to continuously change identities. Both of these issues increase the probability of interacting with a stranger.

In this paper, our contributions are to generalize from the traditional symmetric EPD to the asymmetric transactions of P2P applications, map out the design space of EPD-based incentive techniques, and simulate a subset of these techniques. Our findings are as follows:

- Incentive techniques relying on private history (where entities only use their private histories of entities' actions) fail as the population size increases.
- Shared history (where entities share their histories of other entities' actions) scales to large populations, but requires a supporting infrastructure and is vulnerable to collusion.
- Incentive techniques that adapt to the behavior of strangers can cause systems to converge to complete cooperation despite the existence of zero-cost identities and without centralized identity allocation

### II. MODEL

#### 2.1 Requirements and Assumptions

We have a variety of requirements for modeling cooperative applications like peer-to-peer (P2P) systems. In such a model, universal cooperation should result in optimal overall utility. However, in the absence of incentive mechanisms, individuals who exploit the cooperation of others while not cooperating (*defecting*) should benefit more than users who do cooperate. For example, a P2P file sharing user who downloads from others, but does not share files avoids paying per byte fees to his Internet service provider (ISP) and slowing his own downloads.

Moreover, we assume that all individuals are *strategic*. i.e., they are *rational* users, who will change their behavior (*evolve*) to maximize their own benefit. In reality, there may be other types of individuals, but we concentrate on the strategic majority. The combination of universal cooperation leading to optimal overall utility, an individual incentive to defect, and rational behavior provide the essential tension that results in the tragedy of the commons. A model should have the flexibility to be applied to a variety of peer-to-peer applications. Different applications have different definitions of cooperation and defections and different benefits and costs (the *payoffs*) for users. In the applications we consider, transactions are always between two individuals. In some applications (e.g., P2P file sharing), only one individual (the

server) has the choice of cooperating, while the other (the client) can only receive the cooperation. Furthermore, in many P2P applications, the client cannot trace defections to a particular server. For example, a client in a file sharing network cannot trace his inability to download a file to a specific server who has the file and is refusing to serve it. Finally, we assume that all individuals have the same payoffs.

This is unlikely to be the case in reality since users will value the benefits and costs of services differently. Also, payoffs are likely to change over time as the popularity of services wax and wane.

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

Figure 1: This is general form of a payoff matrix the evolutionary Prisoner's Dilemma R, S, T and P Stand for reward, sucker, temptation, and punishment, respectively.

### 2.2 Evolutionary Prisoner's Dilemma

The traditional evolutionary Prisoner's Dilemma (EPD) model [3] assumes that entities are symmetric: there is no difference between the client and the server in a transaction. These qualities do not satisfy the requirements given above, so, in this section, we describe a generalized form of EPD that allows asymmetry in transactions. This generalized EPD (henceforth referred to as EPD) encompasses the traditional EPD as a special case. EPD consists of players who meet for games. Each player has a score which is initialized to 0. In each game, one player is the client and one player is the server. A player can be a client in one game and a server in another. The client selects the server using a strategy (which also decides actions, see below). The simplest selection algorithm is to select uniformly randomly from the available servers. The client and server each have the choice of cooperating or defecting.

Players decide whether to cooperate or defect using a strategy.

Players observe each other's actions, but not their strategies. A player may maintain a history of other players' actions, which strategies may use. Some typical strategies that exist in current P2P systems are 100% Cooperate and 100% Defect. Given the payoff matrix restrictions described above.

the system requires many 100% Cooperators to drive the system to high overall utility, but the 100% Defectors gain more benefit from the system than the 100% Cooperators.

		Server	
		Allow Download	Ignore Request
Client	Request File	7/-1	0/0
	Don't Request	0/0	0/0

Figure 2: This figure shows the payoff matrix for an application like P2P file sharing or overlay routing.

### III. DESIGN SPACE

Our goal is to design incentive strategies that both drive the system to high overall utility (like 100% Cooperate), while providing more benefit to their players than any defector strategy (like 100% Defect). In this section, we describe the design space of incentive strategies.

**Decision function.** A decision function takes a history of a player's actions and decides whether to cooperate or defect with that player. Our requirements for a decision function are that it can use shared and subjective history (described below), it can deal with untraceable server defections, and it is robust against different patterns of defection. Previously proposed decision functions (e.g., Tit-for-Tat [3] and Image [13]) do not satisfy this criteria. The Reciprocal decision function, where

meets the criteria described above. We show a more complex version of this function in our simulations, but we omit the details for space reasons.

**Private vs. shared history.** Private history is player A's record of player B's actions towards A. Shared history is a record of B's actions towards everyone. Private history does not scale to large population sizes or high turnover because it is only useful when two players have repeat games, but this becomes less likely as population size or turnover increases. Shared history scales better because it only requires that someone has interacted with a particular player. One advantage of private history is that a decentralized implementation is straightforward. However, shared history can also be implemented in decentralized way using a peer-to-peer storage system [5] [11] or by disseminating information to other players in a similar way to routing protocols.

**Strangers.** History assumes that players maintain persistent identities. However, in most P2P systems, identities are zero-cost. This allows the system to grow quickly, but also allows users to continuously change identities to escape the consequences of their past actions. A *stranger* in the system could either be a legitimate *newcomer* or one of these *whitewashers*. We deal with whitewashers by varying a strategy's policy towards strangers.

**Objective vs. subjective reputation.** While shared history is scalable, it is vulnerable to collusion. For example, defecting players can claim that other defecting players cooperated with them. This subverts any strategy in which everyone in the system agrees on the reputation of a player (*objective reputation*). An example is to use the Reciprocative decision function with shared history to count the total number cooperations a player has given to and received from all players in the system. Instead, to deal with collusion, players should compute reputation *subjectively*, where player A weights player B's opinions based on how much player A trusts player B. One example of a subjective algorithm is max-flow [12] [14], which computes the maximum flow between any pair of nodes in a graph. **Selection.** In addition to deciding actions, strategies select players for games. By selecting carefully, a strategy can avoid strangers and known defectors. However, in many P2P systems, a client can only obtain the desired service from a subset of the available servers, thus limiting the benefit of selection.

#### IV. RESULTS

In this section, we use simulation to explore part of the design space described in the previous section. Our simulator implements the model described in Section 2. In all of the simulation scenarios, we start with equal numbers of 100% Cooperators and 100% Defectors and varying numbers of discriminating players using the Reciprocative decision function. In addition, we vary the population size (from 24 to 3000) and the number of games per round. We use 50 generations, which is long enough for one strategy to dominate the system in the simulation scenarios we consider. We use the file sharing payoff matrix described in Section 2. Results using other payoff matrices are similar.

##### 4.1 Model Dynamics

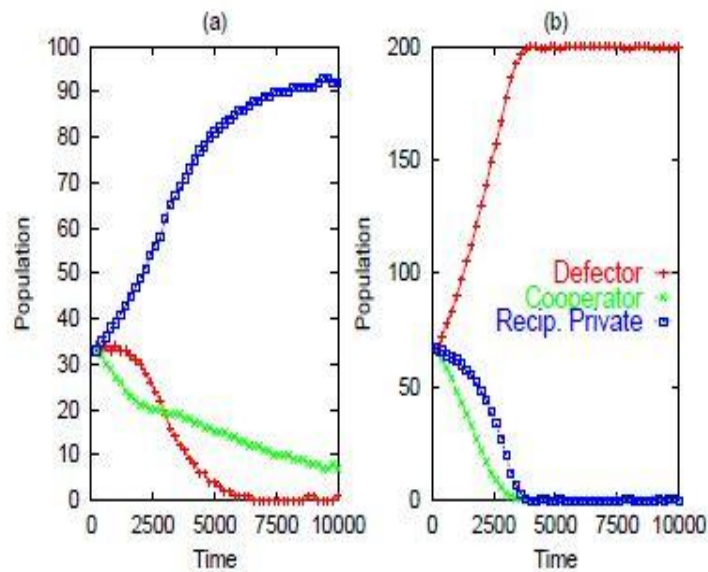


Figure 3: The evolution of strategy populations over time. “Time” the number of elapsed rounds. “Population” is the number of players using a strategy.

However, the convergence of Reciprocative using private history varies depending on the population size, the initial mix of the population and the rate at which players are making transactions (the rounds / generation). A higher initial percentage of the population (right column) or a higher rate of transactions (bottom row) allows Reciprocative with private history to converge at larger population sizes. However, Reciprocative with private history inevitably fails at some point as the population increases. This occurs because it is less likely that a Reciprocative player will have repeat games with the same player as the population size increases. Therefore, such a player using private history is more likely to be taken advantage of by a defector. This allows the defectors to dominate the system and drive cooperation to zero. In contrast, shared history allows players to leverage off of the experiences of others and does scale to large population sizes.

For the large ( 100,000 nodes) P2P systems that are common on the Internet, private history is not sufficient. Shared history is expensive to maintain and vulnerable to collusion, but these results show that it is worthwhile to explore methods of reducing the cost and shielding the vulnerability.

#### V. RELATED WORK

Previous work has examined the incentive problem for Internet applications and specifically when applied to peer-to-peer systems. The DAMD (distributed algorithmic mechanism design) approach focuses on incentive compatibility solutions

to Internet problems in a decentralized manner [7]. Axelrod [3] introduces the Evolutionary Prisoner's Dilemma (EPD) as a model for understanding cooperation. In a simulation environment with many repeated games, persistent identities, and no collusion, Axelrod shows that the Tit-for-Tat strategy dominates. Some researchers [4] [6] show that whitewashing and collusion can have dire consequences for peer-to-peer systems and are difficult to prevent in a fully decentralized system. Our goal is to disincentivize these attacks instead of preventing them. Friedman and Resnick [8] state that punishing all newcomers is inevitable in systems with zero-cost identities. They show that such a system can converge to cooperation only for sufficiently low turnover rates, which our results confirm. To avoid entry fees, they propose the centralized allocation of identities, which are free but unreplaceable. However, this authority is likely to be expensive to maintain, thus shifting from cost for entry to cost for identity allocation. 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. Peers in the GUNet [9] file sharing system deal with this problem by keeping private history about transactions with other peers. We show in Section 4.2 that this does not scale to large numbers of peers.

## REFERENCES

- [1] Kazaa. <http://www.kazaa.com>.
- [2] Limewire. <http://www.limewire.com>.
- [3] AXELROD, R. *The Evolution of Cooperation*. Basic Books, 1984.
- [4] 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).
- [5] DABEK, F., KAASHOEK, M. F., KARGER, D., MORRIS, R., AND STOICA, I. Wide-area cooperative storage with CFS. In *Proceedings of the ACM Symposium on Operating Systems Principles* (Oct. 2001).
- [6] DOUCEUR, J. R. The Sybil Attack. In *Electronic Proceedings of the International Workshop on Peer-to-Peer Systems* (2002).
- [7] 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).
- [8] FRIEDMAN, E., AND RESNICK, P. The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy* 10, 2 (1998), 173–199.
- [9] GROTHOFF, C. GUNet - An Excess Based Economy. *Wirtschaftsinformatik Peer-to-Peer (P2P): Technologies, Architectures, and Applications* (March 2003).
- [10] HARDIN, G. The Tragedy of the Commons. *Science* 162 (1968), 1243–1248.
- [11] KUBIATOWICZ, J., BINDEL, D., CHEN, Y., EATON, P., GEELS, D., GUMMADI, R., RHEA, S., WEATHERSPOON, H., WEIMER, W., WELLS, C., AND ZHAO, B. OceanStore: An Architecture for Global-scale Persistent Storage. In *Proceedings of ACM ASPLOS* (Nov. 2000), ACM.
- [12] LEVIEN, R., AND AIKEN, A. Attack-Resistant Trust Metrics for Public Key Certification. In *Proceedings of the USENIX Security Symposium* (1998), pp. 229–242.
- [13] NOWAK, M. A., AND SIGMUND, K. Evolution of Indirect Reciprocity by Image Scoring. *Nature* 393 (1998), 573–577.
- [14] REITER, M. K., AND STUBBLEBINE, S. G. Authentication Metric Analysis and Design. *ACM Transactions on Information and System Security* 2, 2 (1999), 138–158.