Incentive Mechanisms for Peer-to-Peer Systems

Bin Yu and Munindar P. Singh

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535, USA

{byu, mpsingh}@eos.ncsu.edu

Abstract. Most of the existing research in peer-to-peer systems focuses on protocol design and doesn't consider the rationality of each peer. One phenomenon that should not be ignored is free riding. Some peers simply consume system resources but contribute nothing to the system. In this paper we present an agent-based peer-to-peer system, in which each peer is a software agent and the agents cooperate to search the whole system through referrals. We present a static and a dynamic pricing mechanism to motivate each agent to behave rationally while still achieving good overall system performance. We study the behavior of the agents under two pricing mechanisms and evaluate the impact of free riding using simulations.

1 Introduction

Peer-to-peer (P2P) systems are currently receiving considerable interest in both industry and academia. P2P systems have emerged as a promising way to share files (Napster, Gnutella, and FreeNet), computing resource (SETI@home), and other valuable information, e.g., reputation information [16]. P2P systems have also been studied in academia recently, e.g., CAN [10], Chord [14], and Pastry [11]. These projects study distributed hashing algorithms. Given an object, the algorithms guarantee to locate a peer that has that object. However, most of the present research has been focused on protocol design concerns such as file lookup, data replication, and load balancing. Typically, current approaches don't consider the rationality of each peer and simply assume that the peers will follow the given protocols.

One phenomenon that should not be ignored is free riding. Since users do not benefit directly from sharing files with others, many users choose to decline the requests from others. Free riding is found in many P2P systems but is not punished [4, 9]. For example, in Gnutella, there is a significant amount of free-riding users. Adar and Huberman found that 70% of the Gnutella users did not share any content files and 90% did not answer to any queries from other peers [1]. Uncontrolled or excessive free riding in a P2P network leads to network congestion at some hotspot peers and the degradation of system performance. It is thus important to design some mechanisms that encourage peers to contribute and reduce free riding behavior in the P2P networks.

This paper presents an agent-based peer-to-peer system, e.g., a referral system, in which each peer is a software agent and the agents cooperate to search the whole system through referrals. Agents are rational and self-interested, and so they may not always follow the protocols as the designer expects. Individuals participating in a referral system can contribute in two ways: The first is simply by answering the queries. The second is by actively giving referrals, thus providing the "glue" that holds the system together. In a querying process, the agents could play one of the following three roles:

- *requesters*, who request and obtain answers from the referral systems.
- *providers*, who answer the queries from requesters.
- *intermediaries*, who provide referrals and facilitate interactions among requesters and providers.

The study of referrals is important for the development of agent-based peer-to-peer systems that lack specialized agents such as brokers or facilitators [2]. MINDS and ReferralWeb are two previous approaches for referral systems. MINDS emphasizes learning heuristics for referral generation [5], whereas ReferralWeb focuses on how to bootstrap the referral system [6]. More recently, we focus on the effects of topology dynamics on information flows and consider how to efficiently search large-scale unstructured P2P systems, e.g., social networks, with the help of agents who act only on the basis of local knowledge [17]. However, the problem of *free riding* remains to be addressed. Many agents simply ignore the requests and may not give any answers.

In order to control free riding, we introduce pricing mechanisms into referral systems. We view the referral systems as a strategic game, in which each agent has a utility function over their possible actions. We assume that a *rational* agent plays a *strategy* to maximize its own expected utility. Free-riding is an example of strategy where rational users free ride and consume a resource but do not produce at the same level as their consumption. We study the behavior of agents under two pricing mechanisms and then evaluate the impact of free riding using experiments. Our goal is to design some incentive mechanisms that motivate each agent to behave rationally while still achieving good overall system performance [8, 13].

The rest of this paper is organized as follows. Section 2 provides an overview of peer-to-peer systems and referral systems. Section 3 describes the design of pricing mechanisms and related micropayment protocols. Section 4 presents some experimental results. Section 5 summarizes the relevant literature. Section 6 discusses the main themes and some directions for future research.

2 Agent-Based Peer-to-Peer Systems

The term *peer-to-peer* is a generic label for network architectures where all the nodes offer the same services and follow the same behavior. The topology of peer-to-peer systems could be structured, e.g., CAN [10], Chord [14], and Pastry [11], or unstructured, e.g., Napster, Gnutella, and FreeNet. In this paper we only consider the unstructured P2P systems. There are three main alternatives for the implementation of unstructured P2P systems.

- *centralized indexes:* The best example is Napster. Napster uses a centralized database to index the files each peer has in the system. To look for a file, a peer first sends a request to the database, and then gets a list of other peers who may have the files.

- *Pure P2P:* The best known examples are Gnutella and FreeNet. Both of them have a pure distributed architecture, where there is no centralized database. All the peers in the systems establish a connection with others through request propagation.
- *Hybrid solutions:* Hybrid solutions have recently emerged, for example, FastTrack. FastTrack has some supernodes, which are used for indexing the contents of part of the system and play a major role in the organization of the systems.

2.1 Peer-to-Peer Systems

In P2P systems, a P2P node broadcasts a request to its peers, who propagate the request to their peers, and so on. Messages that are broadcast are labeled by a unique identifier, which is used by the recipient to detect where the message comes from. To reduce the network congestion, all messages are characterized by a given TTL (Time to Live) that defines the scope of searches. On passing through a node, the TTL of a forwarded message is decreased by one. When the TTL reaches zero, the message is dropped.

However, many P2P systems form in an ad-hoc manner and do not consider the interests of the peers and dynamics of the topology. In this paper we present agent-based P2P systems, in which each peer is a software agent. The agent can learn about which of its peers are more effective than others, and optimize the searching process based on its past experience. Next we introduce a class of agent-based peer-to-peer systems, referral systems, in which each peer is an agent, and the agents cooperate to search the whole systems through referrals.

2.2 Referral Systems

Intuitively, in a referral system, each agent maintains a list of its *acquaintances*. A query in natural language specifies what information is being sought. A query from the agent is sent to agents of the selected contacts. An agent who receives a query can decide if it can answer or not (Each agent is associated with a user, who will eventually answer the query.). If not, the agent *may* respond with referrals to others. In referral systems, the requesting agent does not propagate the request to its peers. All referrals are sent back to the requesting agent, who tracks the search process using a graph and adaptively directs or ends the process.

Each agent maintains models of its *acquaintances*. The closest acquaintances are called *neighbors*. An agent sends its query initially only to some of its neighbors. If an agent receives a referral, it may pursue the referral even if the referred party is not already an acquaintance—this is how acquaintances are added. An agent adapts its models of its acquaintances from its interactions with others, e.g., when they ask or answer a query. Each agent is allowed only a small number of neighbors; however, no hard limit is imposed on the number of acquaintances. Periodically, an agent may promote some of its acquaintances to becoming its neighbors and also demote some existing neighbors to make room for the new ones.

Each agent maintains two kinds of models: a *profile* for itself; and an *acquaintance model* for each of its acquaintances. We capture these models via the vector space model (VSM) [12], a classical information retrieval technique. The vectors in VSM are term vectors indicating a weight for each term. In our formulation, the terms correspond to different areas of expertise. The expertise of each agent is modeled as a term vector. Similarly, the query is modeled as a term vector.

In VSM, the similarity between two term vectors is defined as the cosine of the angle between them. We define the similarity between a query and an expertise vector as the cosine of the angle between them, but scaled by the length of the expertise vector. Intuitively, for two agents with expertise in the same direction, the one with the greater expertise is more desirable, whereas the traditional definition would treat them alike.

Definition 1. Given a query vector $Q = \langle q_1, q_2, \ldots, q_n \rangle$ and an expertise vector $E = \langle e_1, e_2, \ldots, e_n \rangle$, the similarity between Q and E is defined as:

$$Q \diamond E = \frac{\sum_{t=1}^{n} q_t e_t}{\sqrt{n \sum_{t=1}^{n} (q_t)^2}}$$

For example, consider a query vector $Q = \langle 0.1, 0.9 \rangle$ and two expertise vectors $E_1 = \langle 0.5, 0.5 \rangle$ and $E_2 = \langle 1, 1 \rangle$. In VSM, E_1 and E_2 are equally similar with the query vector Q, but in our approach, E_2 is better than E_1, since $Q \diamond E_2 > Q \diamond E_1$.

The sociability of an agent reflects its ability to give good referrals. The intuition is that some agents may not be good experts, but may be well connected and may give good referrals. Therefore, the relevance of a neighbor to a given query depends not only on the similarity of the query to the user's expertise, but also on the weight assigned to sociability versus expertise.

Definition 2. The relevance of a query vector Q to P_j is computed as $Q \triangle P_j = (1 - \eta)(Q \diamond E_j) + \eta S_j$, where E_j is the expertise of P_j, S_j is the sociability of P_j, and η is the weight given to sociability.

Our previous work studied the effects of η on the quality of referral systems [18]. We found that a certain emphasis (during learning and querying) on the agents' referring ability improves the quality of the system, but that an overemphasis on referrals at the cost of expertise is not useful. For simplicity, we only consider the case $\eta = 0.3$ here.

Each agent learns its profile and its acquaintance models based on an evaluation of the answers received as well as the referrals that led to them. A *referral graph* encodes how the computation spreads as a query originates from an agent and referrals or answers are sent back to this agent.

Definition 3. A referral r to A_j returned from A_i is written as $\langle A_i, A_j \rangle$, we say A_i is a *parent* of A_j and A_j is a *child* of A_i.

For convenience, we include the initial query among the referrals. This enables us to write a referral chain of length l for a query originating with A_r as $\langle A_r, A_1, \ldots, A_l \rangle$. Then *ancestor* and *descendant* are easily defined based on parent and child, respectively.

The referral chains for a given query induce a directed graph whose root is the originating agent. The *depth* of a referral is its distance on the shortest path from the root. Our algorithms ensure that the graph remains acyclic.

Definition 4. A referral graph $G(Q)$ for a query Q is a rooted directed graph (A_r, Λ, R), where A_r is the requesting agent (root), $\Lambda = \{A_1, A_2, \ldots, A_n\}$ is a finite set of agents (vertices) that includes A_r, $R \subseteq \Lambda \times \Lambda$ is a set of referrals (edges).

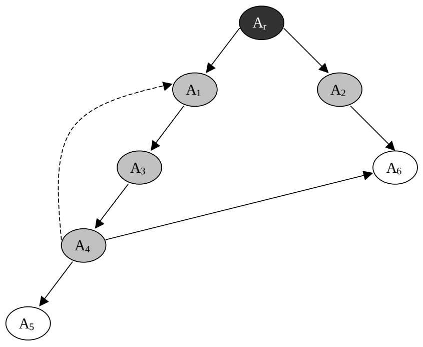

Fig. 1. A referral graph generated from a query. The requesting agent is black; the agents that have been queried are gray; the agent who have not been queried are white.

Definition 5. A referral $r = \langle A_i, A_j \rangle$ is *redundant* for a referral graph (A_r, Λ, R), if and only if $A_i, A_j \in \Lambda$ and A_j is an ancestor of A_i with respect to R.

Clearly, an acyclic referral graph includes no redundant referrals. In the context of Figure 1, a referral $\langle A_4, A_1 \rangle$ would be redundant, since A_1 is an ancestor of A_4. Referral $\langle A_4, A_2 \rangle$ is not redundant, since it introduces no cycles.

3 Mechanism Design

Much of the existing research in P2P systems, including referral systems, assumes that peers or agents will always follow the protocols. However, some agents, representing rational users, may deviate from a designed protocol in order to maximize their outcome. Recently, Shneidman and Parkes advocate mechanism design of P2P systems, in which peers are expected to be rational and self-interested [13]. Feigenbaum and Shenker consider similar problems in distributed algorithmic design. They discuss the challenges of distributed mechanism design in P2P systems and overlay networks with techniques like redundancy and cryptography [3].

We study the mechanism design problem in the context of referral systems. We discuss some micropayment protocols in referral systems, and then study the behavior of agents and the impact of free riding using experiments.

3.1 Types of agents in referral systems

Given a query, some agents may respond unconditionally, and others may respond only if they have some rewards. We categorize the agents in referral systems as one of the following three types.

– **Altruistic:** agents always follow the protocols and give answers or referrals if they can.

- **Rational:** agents play a strategy to maximize their expected outcome.
- **Irrational:** agents do not follow a strategy modeled by the mechanism. Antisocial or malicious agents, for instance, prefer strategies that hurt other agents even when these strategies reduce their own utility.

The altruistic and irrational agents are outside of our discussion. In this paper we only focus on rational agents that can strategize about their behavior.

3.2 Micropayment Protocol

A natural approach is to charge agents for every query and to reward them for every referral or answer. [1] In this section we first describe a simple micropayment mechanism, in which the costs for referrals and answers are fixed for all agents. We then present a more complex protocol, where the costs are dynamic. Suppose,

- α is the cost or reward for one or more referrals given for a query.
- β is the cost or reward for an answer to a query.
- T is the initial budget for each agent, e.g., 500 points.

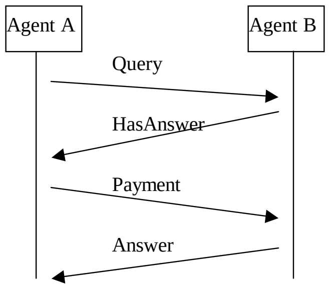

Fig. 2. A referring process involving two agents A and B, where A sends a query to B.

We illustrate the micropayment protocol using two simple examples.

- First, we consider the situation only involving two agents A and B. Agent A sends a query to one of its neighbors B. If agent B finds that it can answer the query, it will answer with a *HasAnswer* message. Agent A will decide if it would like to pay. If A agrees and pays the necessary points, agent B will send the answer to A. In this process,
  - The cost for agent A is β and the balance for agent A becomes $T - \beta$.

[1] Another possible payment model is a flat rate membership fee. However, the flat fees are unrelated to agent's strategies, and may not be helpful for the free riding problem.

    - The reward for agent B is β and the balance for agent B becomes $T + \beta$.

- Second, we consider the situation involving three agents A, B, and C. Agent A sends a query to B, and B finds that it cannot answer the query. However, B has some neighbors who may answer the query from A. B responds with a *HasReferral* message. Agent A will decide if it would like to pay for the referrals. Agent A receives a set of referrals after it pays to B. Suppose one of the referrals leads to agent C and all others lead to dead-ends. C responds with an answer after A pays the points to C.
    - The cost for agent A is $\alpha + \beta$ and the balance for agent A becomes $T - \alpha - \beta$.
    - The reward for agent B is α and the balance for agent B becomes $T + \alpha$.
    - The reward for agent C is β and the balance for agent C becomes $T + \beta$.

Algorithm 1 Constructing a referral graph

```
1: Suppose agent $A_r$ is the requesting agent, set $\Lambda$ is the agents being visited.
2: Initially $\Lambda = \{A_r\}$. For any agent $A_i \in \Lambda$, $A_r$ sends a query to $A_i$.
3: (If $A_i = A_r$, it means that $A_r$ first sends a query to some of its neighbors).
4: if ($A_i$ returns a HasAnswer message) then
5:     $A_r$ pays the points to $A_i$
6:     $A_r$ receives the answer from $A_i$
7: else if ($A_i$ returns a HasReferral message) then
8:     $A_r$ pays the points to $A_i$
9:     $A_r$ receives a set of referrals from $A_i$
10:    For any referral $r = \langle A_i, A_j \rangle$,
11:    if ($A_j \notin \Lambda$) then
12:        $A_r$ appends $r$ to the referral graph
13:        $A_r$ adds $A_j$ into $\Lambda$
14:    else if ($A_j \in \Lambda$) and ($A_j \neq ancestor(A_i)$) then
15:        $A_r$ appends $r$ to the referral graph
16:    else
17:        Ignore referral $r$
18:    end if
19: end if
```

Algorithm 1 presents the process of constructing a referral graph from a set of referrals. For example, in Figure 1, requesting agent A_r sends a query to its neighbors A_1 and A_2. A_1 refers to A_3, who refers to A_4. A_4 refers to A_1, A_5, and A_6. A_2 refers A_6. Suppose A_5 and A_6 claim they have the answer, and A_r pays to both of them. Eventually A_6 returns an answer, but A_5 doesn't. The costs (rewards) for these agents are,

- The cost for A_r is $4\alpha + 2\beta$.
- The reward for each of A_1, A_2, A_3, and A_4 is α.
- The reward for each of A_5 and A_6 is β.

3.3 Dynamic Pricing

A more complex case is that the costs or rewards are dynamic. Since different agents provide different qualities of services and they may place different prices for their referrals and answers. Also, some agents claim they have the answers or referrals, but they may not respond after the requesting agent pays, e.g., A_5 in Figure 1. The requesting agent needs to decide which service it would like to buy, based on the history of responding agents, and the number of agents who can provide the services.

We assume that each responding agent produces results randomized around a certain quality for referrals and answers. But the quality of referrals and answers from different agents may be different. As mentioned in Section 2, each agent has a profile and a set of acquaintance models. Suppose A_r is the requesting agent, and $\{A_1, A_2, \ldots, A_n\}$ are a set of acquaintances of A_r. A_r has two *selling prices* in its profile: α'_{A_r} for one or more referrals (given for a specific query), and β'_{A_r} for an answer. Similarly, for any agent A_i, $1 \leq i \leq n$, A_r has two *reserve prices* in A_i's acquaintance model: α_{A_i} for one or more referrals and β_{A_i} for an answer.

The values of α and β are used as the baselines for both selling prices and reserve prices. For example, at time t_0 (which is local to agent A_r), given a requesting agent A_r and any of its acquaintances A_i,

$$\alpha'_{A_r}(t_0) = \alpha_{A_i}(t_0) = \alpha$$
$$\beta'_{A_r}(t_0) = \beta_{A_i}(t_0) = \beta$$

The selling prices are updated as follows

- The two selling prices will decay with a decaying coefficient ρ at every time interval t, where $0 < \rho < 1$. For example, given a agent A_r, its selling price for a referring service at time $t_0 + t$ is updated as $\alpha'_{A_r}(t_0 + t) = \rho * \alpha'_{A_r}(t_0)$.
- The selling prices of a referring service or an answer will increase with a factor σ when any other agents would like to pay the price, where $1 < \sigma < 2$.

The reserve prices are used to estimate if the selling prices from other agents are reasonable. For example, at time t_i, agent A_r receives a set of sell bids of answers from sellers $\{A_1, A_2, \ldots, A_m\}$. For any seller A_j, the reward for agent A_r is

$$\beta_{A_j}(t_i) - \beta'_{A_j}(t_i)$$

where $\beta_{A_j}(t_i)$ is the reserve price of an answer in the acquaintance model for A_j at time t_i and $\beta'_{A_j}(t_i)$ is the selling price of an answer in the profile of agent A_j.

Similarly, for referring services, the requesting agent A_r computes the reward as

$$\alpha_{A_j}(t_i) - \alpha'_{A_j}(t_i)$$

Given a set of referring services or answers, the requesting agent will choose the services from the highest to the lowest rewards. After the requesting agent receives a service, it or its user will evaluate the quality of the service and revise the reserve price for the service. The reserve prices are updated as follows at time t_i if the agent is satisfied with the service from agent A_j,

$$\alpha_{A_j}(t_i) = \alpha_{A_j}(t_i) + \omega_1 \text{ (for referring services)}$$
$$\beta_{A_j}(t_i) = \beta_{A_j}(t_i) + \omega_2 \text{ (for answers)}$$

Otherwise,

$$\alpha_{A_j}(t_i) = \alpha_{A_j}(t_i) - \omega_1 \text{ (for referring services)}$$
$$\beta_{A_j}(t_i) = \beta_{A_j}(t_i) - \omega_2 \text{ (for answers)}$$

where $0 < \omega_1 < \alpha$ and $0 < \omega_2 < \beta$.

4 Experimental Results

Our experiments are based on an extension of a simulation testbed previously developed for information access [18]. The experiments involve between 100 and 500 agents. Each agent is modeled in terms of its *interest* (describing the services it is interested in purchasing) and its *expertise* (describing the services it is able to offer). Both interest and expertise are captured as terms vectors of dimension 5.

The agents are limited in the number of neighbors they may have, here 4. The length of each referral chain is limited to 4. Moreover, we introduce a probability φ between 0 and 1 to model any free riding agents A_i. Agent A_i will generate an answer from its *expertise* vector upon receiving a query with the probability φ even when there is a good match between the query and its expertise vector.

In each simulation cycle, we randomly designate an agent to be the requester. An agent may query some of its neighbors. When an agent receives a query, it may answer the query based on its expertise vector, or may give a referral to some of its neighbors. The originating agent collects all possible referrals, and continues the process by following some of the suggested referrals. Each agent may keep track of certain acquaintances. In our simulation, we allow 12 acquaintances. Periodically, each agent decides which of its acquaintances are dropped and which are promoted to neighbors (a subset of acquaintances).

We initialize the network of agents in the following manner. Following Watts and Strogatz [15], we begin from a ring but, unlike them, we allow for edges to be directed. We use a regular ring with 100 nodes, and 4 out-edges per node (to its neighbors) as a starting point for the experiment.

The initial budget for each agent is 500. The baselines for prices of a referring service and an answer are 1 and 10, respectively. Other parameters for dynamic pricings are defined as follows,

– Decaying coefficient $\rho = 0.9$ for every 100 cycles.
– Factor $\sigma = 1.1$.
– Other factors $\omega_1 = 0.1$, $\omega_2 = 1$ (We choose the values of ω_1 and ω_2 in the same ratio as α and β).

Note that all these parameters are fixed and equal for all agents. An answer is good if and only if the similarity value between the query and the answer is above 0.2.

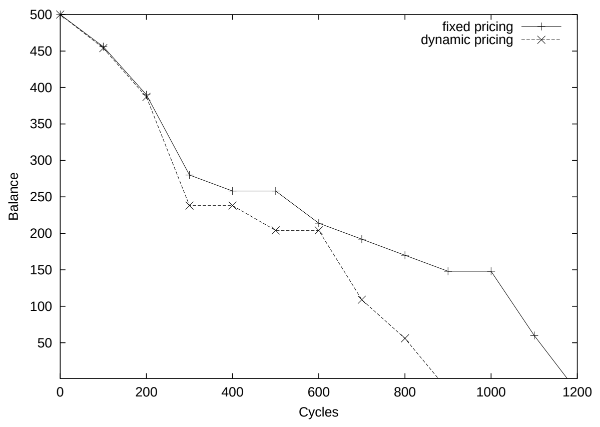

Fig. 3. Balances of the free riding agent under different pricing mechanisms

4.1 Balance of the Free Rider

We suppose, of the total 100 agents, only one agent is a free rider. Its responding probability φ is zero. Figure 3 shows the balance of the free riding agent under fixed and dynamic micropayment protocols. As intuitively expected, the free riding agent cannot survive under either micropayment protocol. The balance of the free riding agent becomes zero after 1200 cycles under fixed pricing mechanism and 900 cycles under dynamic pricing mechanism. The agent who runs out of its budget has to purchase more points with money. In a sense, no one can free ride any more, because they have to pay for the services they receive from others.

4.2 Prices for High-Quality Services

Our second experiment studies the selling prices of referrals and answers for an expert agent, where each dimension of its expertise vector is initialized as 1. For example, the selling price of answers (from the expert agent) increases from 10 to about 55 and the selling price of referrals increases from 1 to 15 after 1000 cycles (Figure 4). A consequence of dynamic pricing is that the requesting agents have to pay more for the high-quality services. The prices will help to adjust the traffic at these high-quality service providers. Note that the selling prices for answers and referral from the expert agent may decrease if no agents are willing to pay the prices.

5 Related Work

Golle *et al.* first use a game theoretic approach to analyze the free riding problem in peer-to-peer file sharing systems [4]. They analyze equilibria of user strategies under several micropayment mechanisms. Our micropayment protocol with fixed pricing is similar to theirs. More recently, Ramaswamy and Liu use utility functions to measure

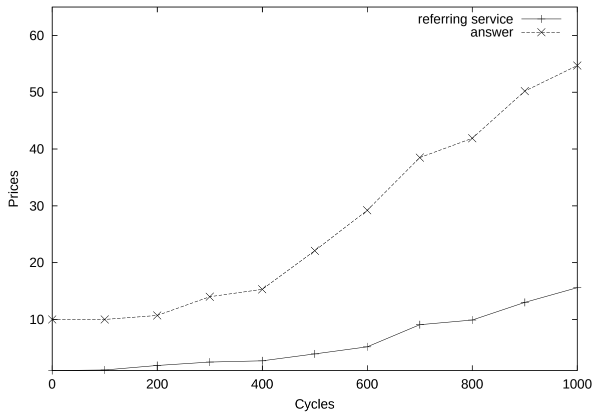

Fig. 4. Selling prices of high-quality services

the usefulness of peers, e.g., the number of files, the total size of the data, and the popularity of the files, and describe a utility based scheme to control free riding in peer-to-peer systems [9]. Both of the above approaches study the free riding problem in P2P file sharing systems, while we focus on the referral systems, in which the prices of services can be either fixed or dynamic.

Shneidman and Parkes discuss the notions of rationality and self-interest in P2P systems [13]. Similar ideas can also be found in Distributed Algorithmic Mechanism Design (DAMD) [3]. Shneidman and Parkes highlight some open problems in DAMD and specially P2P systems, e.g., computational complexity of mechanisms and effects of mechanism design on network topology formation.

Pricing or micropayment is only one incentive mechanism. Krishnan *et al.* propose other mechanisms to reduce the problem of free riding in P2P systems [7]. They develop some non-priced incentives to encourage efficient behavior in P2P users. Some examples include delay time (e.g., users who share more content with the system have higher priority), network membership (e.g., removing non-sharing members from the systems), or peer ratings of content providers. However, empirical analysis is needed to measure the impact of these mechanisms.

6 Conclusion

This paper examines the problem of free riding in agent-based peer-to-peer systems, especially referral systems. We introduce two classes of micropayment protocols and analyze the strategies of agents under these protocols. Our paper only provides a preliminary study of mechanism design in referral systems. For example, we simply assume that the qualities of services are consistent for each agent. Also, we don't consider the topology of referral graphs and its effects on dynamic pricing. In future work, we plan to focus on these problems and develop more efficient and incentive compatible mechanisms for referral systems.

Acknowledgements

This research was supported by the National Science Foundation under grant ITR-0081742. We are indebted to the anonymous reviewers for their helpful comments.

References

1. E. Adar and B. Huberman. Free riding on Gnutella. *First Monday*, 5(10), 2000.
2. K. Decker, K. Sycara, and M. Williamson. Middle-agents for the Internet. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 578–583, 1997.
3. J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the Sixth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, 2002.
4. P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge. Incentives for sharing in peer-to-peer networks. In *Proceedings of the Second International Workshop on Electronic Commerce*, pages 75–87, 2001.
5. M. N. Huhns, U. Mukhopadhyay, L. M. Stephens, and R. D. Bonnell. DAI for document retrieval: The MINDS project. In M. N. Huhns, editor, *Distributed Artificial Intelligence*, pages 249–283. Pitman/Morgan Kaufmann, London, 1987.
6. H. Kautz, B. Selman, and M. Shah. The hidden Web. *AI Magazine*, 18(2):27–36, 1997.
7. R. Krishnan, M. D. Smith, and R. Telang. The economics of peer-to-peer networks, 2002. working paper, Carnegie Mellon University.
8. M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, MA, 1994.
9. L. Ramaswamy and L. Liu. Free riding: a new challenge for peer-to-peer file sharing systems. In *Proceedings of Hawaii International Conference on Systems Science 36*, 2003.
10. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, pages 161–172, 2001.
11. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18nd IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001.
12. G. Salton and M. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
13. J. Shneidman and D. Parkes. Rationality and self-interest in peer-to-peer networks. In *Proceedings of Second International Workshop on Peer-to-Peer Systems*, 2003.
14. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, 2001.
15. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.
16. B. Yu and M. P. Singh. An evidential model of distributed reputation management. In *Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 294–301, 2002.
17. B. Yu and M. P. Singh. Searching social networks. In *Proceedings of Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 65–72, 2003.
18. B. Yu, M. Venkatraman, and M. P. Singh. An adaptive social network for information access: Theoretical and experimental results. *Applied Artificial Intelligence*, 17(1):21–38, 2003.