

An Incentive Driven Lookup Protocol For Chord-Based Peer-to-Peer (P2P) Networks ^{*}

Rohit Gupta and Arun K. Somani

Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011, USA
{rohit, arun}@iastate.edu

Abstract. In this paper we describe a novel strategy for carrying out lookups in Chord-based peer-to-peer (P2P) networks, wherein nodes are assumed to behave selfishly. This is in contrast to the traditional lookup schemes, which assume that nodes cooperate with each other and truthfully follow a given protocol in carrying out resource lookups. The proposed scheme also provides efficient and natural means for preventing free-riding problem in Chord without requiring prior trust relationships among nodes. In addition, we evaluate the performance of Chord for providing routing service in a network of selfish nodes and prove that it has good structural properties to be used in uncooperative P2P networks.

1 Introduction

Almost all the current research in P2P systems is based on a cooperative network model. It is generally assumed that although there can be rogue nodes in a system, most of the nodes are trustworthy and follow some specific protocol as suggested by the network designer. We believe that such assumptions do not always hold good in large-scale open systems and have to be done away with in order to make P2P systems reliable, robust, and realize their true commercial potential. Moreover, it has been pointed out that free-riding is one of the most significant problems being faced by today's P2P networks [2].

We consider a Chord [1] based P2P network and describe a novel strategy for carrying out lookups in such networks. Our proposed scheme provides an efficient and natural means for preventing free-riding problem in Chord without requiring prior trust relationships among nodes. Therefore, it incurs low overhead and is highly robust, and unlike other schemes it does not rely on any centralized entity or require specialized trusted hardware at each node. The protocol proposed here is essentially an *incentive driven lookup* protocol that ensures that reward received by intermediate nodes and resource provider is maximized by following the protocol steps. It is in contrast to other lookup schemes, which assume that nodes cooperate with each other in finding data and faithfully follow a given protocol for carrying out resource lookups irrespective of whether they are currently overloaded or not, for example.

^{*} The research reported in this paper is funded in part by Jerry R. Junkins Chair position at Iowa State University.

We evaluate the performance of Chord for providing routing service in a network of selfish nodes. We show that in a large network, unless nodes have privilege information about the location of network resources, following Chord is a good strategy provided that everyone else also follow the Chord protocol.

The paper is structured as follows. Section 2 is on related work, Section 3 describes the network model. Section 4 gives a detailed description of the proposed lookup protocol. Section 5 presents a resource index replication strategy that is useful in dealing with selfish nodes in Chord. Section 6 explains why Chord is a good protocol to be used in a network with selfish nodes. We conclude the paper in Section 7.

2 Related Work

The need for developing protocols for selfish agents (nodes) in P2P systems has often been stressed before (see [3, 4]). The research in [5, 6] provides solution to avoid free-riding problem in P2P networks. The basic approach in all of these is to make sure that nodes indeed share their resources before they themselves can obtain services from a network. Also, most of these solutions rely on self-less participation of groups of trusted nodes to monitor/police the activities of individual nodes, and ensure that everyone contributes to the system.

To the best of our knowledge, none of the existing solutions that deal with the problem of free-riding in P2P networks also address the more basic question of why nodes would route messages for others. Since these nodes belong to end users, they may in order to conserve their bandwidth and other resources, such as buffer space, memory etc., may drop messages received for forwarding.

The problem of selfish routing has been encountered and addressed in the context of mobile ad-hoc networks (see [7, 8]). Some of these proposals can also find application in P2P networks.

3 Network Model

The model of network assumed here is a Chord [1] based P2P network. The nodes are assumed to be selfish. By *selfish* we mean that nodes try to maximize their profits given any possible opportunity. The profit from a transaction (or an activity) is equal to the difference between the reward that a node earns and the cost that it incurs by participating in the transaction. The reward can be anything that is deemed to have value, the possession of which adds to a node's utility.

An example of a transaction is a lookup process, i.e. the process of searching for and downloading a desired resource object. The cost in the form of bandwidth, memory etc. that a node x incurs by participating in a transaction is referred to as its marginal cost (MC_x). The cost incurred by server S (and also the intermediate nodes) increases in proportion to the amount of traffic it is handling and any request offering less than its current MC_S value is not fulfilled.

We assume that for each resource there is a single server, i.e. caching and replication of data does not take place. Nodes that store the index of a resource are called

the *terminal* nodes for that resource. For resource R these nodes are denoted by T_{R_i} , $\forall i \in \{1, \dots, k\}$, where k is the index replication factor. The terminal nodes maintain a mapping (i.e. an index) from the resource name, represented by R , to the IP address of the server that provides the resource. The terminal nodes are the Chord successors of the mappings of a resource onto the Chord network. The method by which these mappings are determined is explained in Section 5.

Unless otherwise specified, all message communication is assumed to provide message non-repudiation. Our protocol relies on message non-repudiation to ensure that nodes do not go back on their *commitment* as suggested by the content of the messages sent by them. We assume that there is a mechanism in place to punish nodes if it can be proven that they did not fulfill their commitments.¹

4 Incentive Driven Lookup Protocol

We now describe how routing of lookup messages is performed when nodes behave selfishly and how prices for resources are set with minimum additional overhead on the system. To simplify our discussion, we take an example of a lookup process and see how it is carried out under the given protocol.

4.1 Parallel Lookup Towards the Terminal Nodes

The client (C) before initiating the lookup process estimates its utility (U_R^C) of the resource (R) to calculate the maximum price that it can offer for the resource. Since the locations of the resource indices can be calculated by using the same mechanism as used by the server (S) to store them, C sends a separate lookup message towards each of them. Together these parallel lookup messages can be considered as constituting a single lookup process and the routing of an individual lookup message is done using the Chord routing protocol.

Each lookup message Msg_{lookup} contains the following information, as included by the client - address of one of the k terminal nodes (T_{R_i}), the resource ID (R), the maximum price offered (P_C), the marginal cost (MC_{total}), the request IDs ($Reqid_{private}$ and $Reqid_{public}$).

$Reqid_{public}$ identifies the lookup process such that S (and intermediate nodes) on receiving multiple lookup messages knows that the messages pertain to the same lookup process. Thus, the same value of $Reqid_{public}$ is included in all the lookup messages. On the other hand, a unique value of $Reqid_{private}$ is included in each of the lookup message. In Section 4.3, we illustrate the significance of $Reqid_{private}$. MC_{total} value is the sum of C 's marginal cost MC_C and the marginal cost of the next hop neighbor (also called the successor) to which the message is forwarded.² C before sending the

¹ In an enterprise computing environment there might be a central authority one can report to in order to identify and punish the cheating node. For large-scale open systems one can use reputation mechanisms to ensure that cheating nodes are accurately identified and isolated from receiving services.

² The term "successor" as used here is the same as used in the description of the Chord protocol, where it referred to the node which immediately succeeds an ID value in a Chord ring or

lookup message inquires its successor about its marginal cost. The received value is added by C to its own marginal cost and stored in MC_{total} . Likewise, each intermediate node on receiving the lookup message updates the MC_{total} value by adding to it the marginal cost of its successor.

Intermediate nodes for all the lookup requests route the received lookup messages to the next hop neighbors and this process continues till the messages reach the desired terminal nodes. Since the terminal nodes store the IP address of S , they contact S in order to obtain the resource. S receive k such requests and from the $Reqid_{public}$ values knows that all the requests pertain to the same lookup process. S then holds a second price sealed-bid auction (also called Vickrey auction [9, 3]) with all the terminal nodes as the bidders. S provides the resource to the terminal node that offers it the highest price.

4.2 Bidding for the Resource By the Terminal Nodes

In Vickrey auction, the highest bidder wins the auction, but the price that it has to pay is equal to the second highest bid. Vickrey auction has several desirable properties, such as existence of truth revelation as a dominant strategy, efficiency, low cost etc. Vickrey auction in its most basic form is designed to be used by altruistic auctioneers, which are concerned with overall system efficiency or social good as opposed to self-gains. Self-interested auctioneer is one of the main reasons why Vickrey auction did not find widespread popularity in human societies [10].

Since S behaves selfishly and tries to maximize its own profit, the auction process needs to ensure the following.

- Selecting the highest bidder is the best strategy for S .
- The price paid by the highest bidder is indeed equal to the second highest bid, i.e. S should reveal true second highest bid to the highest bidder.
- Collusion among S and the bidders should not be possible.

In view of the above requirements, we provide a two-phase secure Vickrey auction protocol.

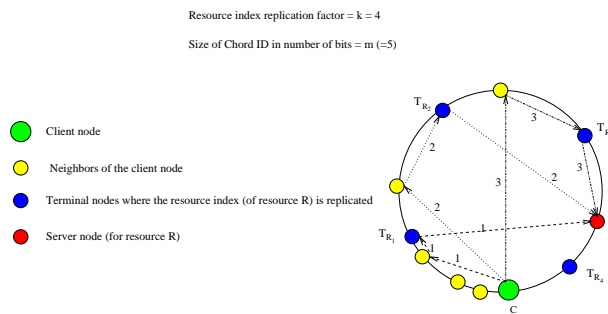


Fig. 1. Lookup message propagation in the *incentive driven lookup* protocol.

In summary, the incentive driven lookup strategy involves sending lookup messages to all the terminal nodes of a resource, such that at most one message is sent out for network. Also, the term "predecessor" refers here to a previous hop node along the lookup path.

all the terminal nodes that go through the same next hop neighbor (the terminal nodes selected is one which is closest to that neighbor). For example, in Fig. 1, C sends a single lookup request message towards T_{R_3} instead of sending towards both T_{R_3} and T_{R_4} . Due to the nature of the Chord routing protocol, with high probability, the number of hops required to go from C to T_{R_3} is less than or equal to that required for going to T_{R_4} . Therefore, the number of terminal nodes that are contacted during a lookup process may be less than the total number of terminal nodes for a resource. However, for simplicity, we assume that all the terminal nodes for a resource are contacted and participate in the lookup process.

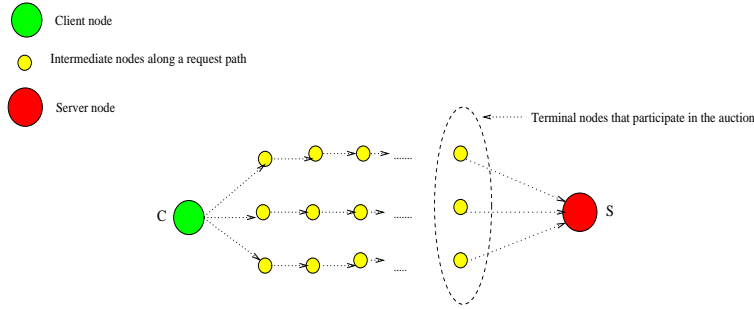


Fig. 2. Formation of request chains due to the propagation of lookup requests.

Fig. 2 depicts an equivalent representation of Fig. 1 and shows different request chains that are formed due to the parallel lookup process. The request chain containing the highest bidder, i.e. the winning terminal node, is called the winning request chain (WRC). In subsequent discussion, we denote the highest and second highest bids by M_1 and M_2 , respectively. The price offered by a terminal node to S is equal to $P_C - MC_{total}$. The amount of profit made by the WRC is equal to $M_1 - M_2$. This profit is shared among the nodes of the WRC in proportion to their marginal costs, i.e. nodes with higher marginal costs get a higher proportion of the total profit, and vice versa.

4.3 Secure Vickrey Auction to Determine Resource Prices

S employs a two-phase Vickrey auction to select the highest bidder and determine the price at which the resource is provided. In the first phase, the bidders send encrypted copies ($E(randKey_i; b_i)$) of their bids in message Msg_{bid} to S . Here $E(randKey_i; b_i)$ is the encryption of bid value b_i of terminal node T_{R_i} using a randomly chosen secret key $randKey_i$. Each message Msg_{bid} also includes $Reqid_{public}$ value received by a terminal node, so that S can determine that the bids pertain to the same lookup process. The received encrypted bids are sent by S back to all the bidders in message $Msg_{bid-reply}$. Since after receiving $Msg_{bid-reply}$, the bidders have encrypted copies of all the bids (total k such bids), S is unable to (undetectedly) alter existing or add fake bids.

In the next and last phase of the auction, each bidder after receiving the message $Msg_{bid-reply}$, sends its secret key in message Msg_{key} to S . The received key values are now sent by S back to all the bidders in message $Msg_{key-reply}$. At the end of this phase, S and all the bidders are able to open the encrypted bids and find out about the highest and second highest bids.

S then sends a message Msg_{cert} to the winning terminal node, denoted by T_{RWRC} , certifying that it has won the auction. The received certificate is forwarded along the reverse path, i.e. opposite to that followed by the lookup request, till it reaches C . C then finds out that the resource has been looked up and is available at a price within its initial offer of P_C . Msg_{cert} contains the following information - the highest bid M_1 , the second highest bid M_2 , the total marginal cost MC_{total} (received by S in Msg_{bid}), and the IP addresses and Chord IDs (each Chord ID is represented by m number of bits, which is also the size of the routing table in Chord) of all the terminal nodes that participated in the auction. The terminal nodes are contacted by C to verify the auction results, and the $Reqid_{private}$ values initially sent in the lookup messages are used to authenticate the terminal nodes. (Note that $Reqid_{private}$ values are not sent by the terminal nodes to S in Msg_{bid}). In [11] we provide a detailed analysis of the robustness of the proposed lookup protocol, including the various threat models it is designed to withstand.

The information in messages Msg_{cert} and Msg_{lookup} allow the intermediate nodes, including T_{RWRC} , to calculate their reward for being part of the WRC .³ The knowledge of the auction result also enables C to determine the price it finally has to pay for R . The calculation of exact payoff values are discussed below.

4.4 Rewarding Nodes of the WRC

Msg_{cert} includes the total marginal cost value MC_{total} of all the nodes in the WRC . This information along with the highest and second highest bids determines each WRC node's payoff. For example, node x 's payoff Pay_x is calculated as follows.

$$Pay_x = MC_x + \left(\frac{MC_x}{MC_{total}} * (M_1 - M_2) \right) \quad (1)$$

The amount received by S is equal to M_2 ($> MC_S$). The profit share of C , i.e. the portion of its initial offer that it saves or gets to keep, is similarly calculated as given below.

$$Profit_C = \left(\frac{MC_C}{MC_{total}} * (M_1 - M_2) \right) \quad (2)$$

C after keeping its profit share gives the remaining, i.e. $U_R^C - Profit_C$, to its successor. The successor node in turn after keeping its payoff gives the remainder to its successor, and so on.

5 Resource Index Replication

The proposed resource pricing scheme utilizing Vickrey auction is based on competition among different chains of nodes attempting to forward the lookup request and delivering the resource back to the client. Higher the competition among the nodes is (i.e. more disjoint the request chains are), higher is the robustness of the pricing scheme.

³ The possession of messages Msg_{cert} and Msg_{lookup} serves as a contract between a node and its predecessor regarding the reward that it is entitled to receive (from the predecessor).

If normal Chord index replication is used, i.e. storing the resource index values at the k Chord successors of the ID where the resource hashes to, then with high probability lookups to all these replicas pass through a single (or a small group) node. Such a node can easily control the lookup process and charge arbitrarily high payoffs for forwarding the requests. To avoid such a monopolistic situation and ensure fair competition in setting prices, we propose that resource indices be replicated uniformly around the Chord network at equal distances from each other. In other words, resource Chord ID mappings should span the entire Chord ID space; this ensures that the lookup paths to different index replicas are maximally disjoint, and are not controlled by any single or a group of small nodes. Below we give a mechanism for determining the location for storing index replicas in the network.

If resource R hashes to Chord ID RI (i.e. the output of the hash function, whose input ID is R , is RI),⁴ then the k resource index replicas map to the following Chord IDs.

$$RI_{R_i} = (RI + \frac{2^m}{k} * (i - 1)) \bmod(2^m), \forall i \in \{1, \dots, k\} \quad (3)$$

The index values are then stored at the Chord successors (the terminal nodes) of the IDs represented by $RI_{R_i} \forall i \in \{1, \dots, k\}$. The intent of replication in Chord is to simply obtain fault-tolerance, while in our protocol the intent is to obtain both fault-tolerance as well as fair pricing. Uniformly spacing the resource index ensures that the lookup paths for different index copies are as node disjoint as possible. This is evident from the results of Fig. 3, where we find that the replication strategy described above decreases the probability that the same nodes are included in multiple paths to reach the index replicas. Similar results would be obtained for a network of any size and any replication factor.

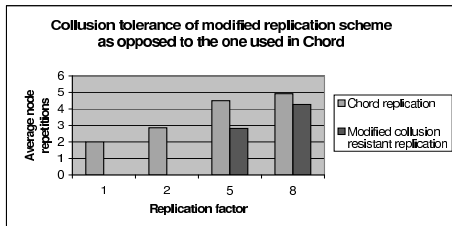


Fig. 3. Average number of repetitions of intermediate nodes that appear in multiple lookup paths to the resource index replica copies. Size of the network, $N = 500$.

6 Selfish Network Topology

So far we have assumed that nodes form a Chord network and the lookup messages are forwarded in accordance with the Chord routing protocol. Now we investigate how correct is the assumption that nodes truthfully follow the Chord protocol. Since nodes

⁴ The hash function used for computing resource Chord ID mapping is the same as that used for determining Chord IDs of the nodes.

are selfish and join a network in order to obtain resources and maximize their profits, the manner in which they select neighbors has a bearing on how successful they are in achieving these goals. This argument definitely holds true for our protocol, since intermediate nodes take their *cut* (equal to their marginal costs) before forwarding a lookup request. Thus, fewer intermediate nodes generally translate to higher profits for the client.

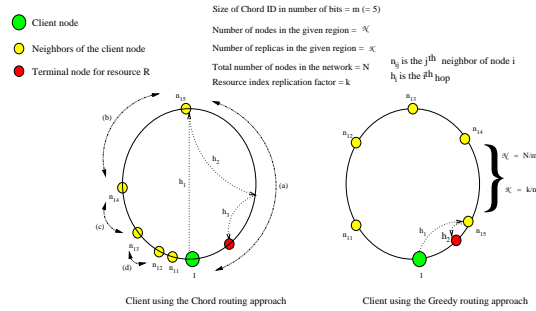


Fig. 4. Comparison of Chord and the greedy routing approach.

A node can make higher profits by being close to terminal nodes of as many different resources that it requires as possible. However, if the location of those terminal nodes is not known beforehand, then it might be advantageous for a node to greedily choose neighbors around the Chord network distributed at equal distances from each other. This strategy seems appropriate, especially since the resource indices are also uniformly distributed around the network. Consider the network shown in Fig. 4 in which node I fill the m ($= 5$) entries in its routing table as per the greedy routing approach instead. So if node I needs to send a message to the terminal node for resource R , it can do it in two hops as opposed to three hops required using Chord. In general, one can see that in at least half of the cases, when the resource to be looked up has Chord ID mapping in region (a), the greedy routing scheme guarantees that the number of hops required to reach the corresponding terminal node is less than (or at most equal to) what is required by Chord. (This assumes that the network size, N , is large and the nodes are uniformly distributed around the Chord network.) Even for the other regions, the greedy approach appears to perform comparably to Chord. This is because node I always first sends a lookup message to its neighbor that is closest (and with lower Chord ID) to the resource Chord ID mapping, and from there on the message is routed using the Chord protocol.

From the above discussion it appears that nodes do not have motivation to follow the Chord protocol, and can make higher profits by utilizing the fact that other nodes follow Chord. In such a scenario the whole routing service would break down, i.e. instead of $O(\log N)$ routing provided by Chord, $O(N/k)$ hops would be required due to the resulting sequential search for the resource indices. However, minimization of the number of routing hops by the greedy approach when everyone else follow Chord is not always correct. We prove below that in a large network, on average the performance of greedy routing approach is no better than that provided by Chord.

Theorem 1. *In a large network, on average the greedy routing approach for any node (say node I in Fig. 4) requires the same number of hops as that required by the Chord routing approach.*

Proof. For the ease of discussion, we assume that the routing table size is fixed for all the nodes in both the approaches and is equal to m (same as it is in Chord).

In Chord, the average number of hops required to reach any node from a given node is $1/2 * (\log N)$. Therefore, the average number of hops required to reach any one of the given set of n nodes (with Chord IDs in the range $0 - 2^{m-1}$) is $\frac{1}{2} * \log(\frac{N/2}{n/2}) = \frac{1}{2} * \log(\frac{N}{n})$. These n nodes are assumed to be located at equal distances from each other. Using these results we obtain the following values.

Average number of hops required by the greedy routing approach to reach one of the k terminal nodes: The neighbors (i.e. the entries in the routing table) of node l in this case completely span the entire Chord ID space, i.e. they are uniformly located at equal distances from each other around the Chord network. Therefore, node l requires the same number of average hops to reach any of the terminal nodes. (The client first sends the request to its neighbor, which then follows the Chord routing protocol to further route the request.)

Thus, the average number of hops taken by the greedy routing approach to reach any resource index replica are given as follows.

$$(1 + \frac{1}{2} * \log(\frac{N}{k})) \quad (4)$$

Average number of hops required by the Chord routing protocol to reach one of the k terminal nodes: Now we calculate the average number of hops required to reach a resource index replica when node l (and everyone else) follows the Chord protocol. It will be $(1 + \frac{1}{2} * \log(\frac{N}{k}))$ when the terminal node is located in region (a), $(1 + \frac{1}{2} * \log(\frac{N}{k}))$ when in region (b)⁵, and so on (up to m such terms).

Therefore, the total average hops needed to reach any of the resource index replicas are given as follows:

$$(1 + \frac{1}{2} * \log(\frac{N}{k})), \quad (5)$$

which is same as the number of hops given by Equation 4.

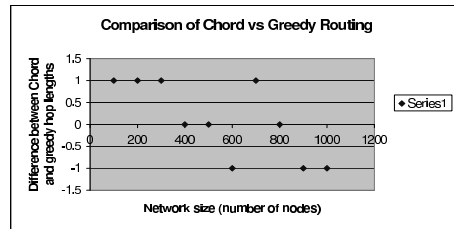


Fig. 5. Comparison of Chord and the greedy routing approaches with regards to the hop-length.

The results in Fig. 5 from our simulations also confirms the fact that a node cannot benefit by selecting the greedy routing strategy. Fig. 5 gives the difference in the observed number of hops when greedy routing is used as opposed to normal Chord routing. We did simulations for varying number of total nodes and averaged the number

⁵ $(1 + \frac{1}{2} * \log(\frac{N/4}{k/4}))$.

of hops for several lookups performed for each network size. As can be seen, there is a difference of at most one hop in the two routing strategies and this is true for both small as well as large network sizes. Thus, a node does not gain an advantage by not following Chord.

The following lemma follows directly from this result.

Lemma 1. *If others in a large network follow the Chord protocol, it is a good strategy to do the same in order to maximize one's payoff.*

7 Conclusion

In this paper we have presented an *incentive driven lookup* protocol for searching and trading resources in Chord-based P2P networks. Our proposed protocol takes selfish behavior of network nodes into account. We used Vickrey auction for setting resource prices in P2P networks and described how it can be implemented in a completely distributed and untrusted environment.

We also investigated the applicability of Chord network topology in forming connectivity among selfish nodes. We proved that in the absence of privilege network information, the best strategy for a node is to follow Chord, provided that everyone else also follows the Chord protocol.

References

1. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. *In Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.
2. A. Eytan, and A. H. Bernardo. Free Riding on Gnutella. *First Monday*, vol. 5, No. 10, Oct. 2000.
3. N. Nisan. Algorithms for Selfish Agents: Mechanism Design for Distributed Computation. *In Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, volume 1563, Springer, Berlin, pages 1-17*, 1999.
4. D. Geels, and J. Kubiawicz. Replica Management Should Be A Game. *In Proceedings of the SIGOPS European Workshop*, 2002.
5. V. Vishumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. *In Proceedings of the 2003 Workshop on Economics of Peer-to-Peer Systems, Berkeley CA*, 2003.
6. S. M. Lui, K. R. Lang, and S. H. Kwok. Participation Incentive Mechanism in Peer-to-Peer Subscription Systems. *In Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, vol. 9, January 2002.
7. S. Zhong, J. Chen, and Y. R. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. *In Proc. of IEEE INFOCOM*, vol. 3, pages 1987-1997, March 2003.
8. L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad-hoc networks. *In Proc. of ACM Journal for Mobile Networks (MONET), special issue on Mobile Ad Hoc Networks*, summer 2002.
9. W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, pages 8-37, 1961.
10. T. Sandholm. Limitations of the Vickrey Auction in Computational Multiagent Systems. *In Proceedings of the 2nd International conference on Multi-Agent Systems*, pages 299-306. Kyoto, Japan, December 1996.
11. Technical Report. http://ecpe.ee.iastate.edu/dcnl/DCNLWEB/Publications/pub-research_tech-rep.htm