A Pricing Strategy For Incentivizing Selfish Nodes To Share Resources In Peer-to-Peer (P2P) Networks

Rohit Gupta and Arun K. Somani
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
E-mail: {rohit, arun}@iastate.edu

***Abstract*— In this paper we describe a novel pricing strategy for carrying out lookups and obtaining data in peer-to-peer (P2P) networks with selfish nodes. Both the resource provider and intermediate nodes that assist in routing of lookup messages are appropriately compensated so as to cover their cost of providing service. This is in contrast to the traditional lookup schemes, which assume that data is freely available, and intermediate nodes selflessly cooperate and truthfully follow a given protocol in carrying out resource lookups. The proposed scheme provides efficient and natural means to prevent free-riding problem in P2P networks, and does not require prior trust relationships among nodes. Moreover, unlike other schemes it does not rely on any centralized entity or require specialized trusted hardware at each node.**

I. Introduction

Almost all the current research in peer-to-peer (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 to realize their true commercial potential. Moreover, it has been pointed out that free-riding, whereby only few altruistic nodes share their resources, is one of the most significant problems being faced by today's P2P networks [1]. Some solutions exist to tackle this problem, but they suffer from one of the following drawbacks - they are either too heavy-weight and expensive (for example, require trusted hardware), or depend on some trusted groups of nodes (or a trusted centralized entity) to police the network and keep the free-riders in check. Trust relationships are, however, difficult to establish in Internet-based P2P settings.

In this paper we describe a novel pricing strategy for carrying out lookups and obtaining data in P2P networks with selfish nodes. Both the resource provider and intermediate nodes that assist in routing of lookup messages are appropriately compensated so as to cover their cost of providing service. This is in contrast to traditional lookup schemes, which assume that data is freely available, and intermediate nodes cooperate and truthfully follow a given protocol in carrying out resource lookups irrespective of whether they themselves are currently overloaded or not, for example. The proposed scheme provides efficient and natural means to prevent free-riding problem in P2P networks and does not require prior trust relationships among nodes. Moreover, unlike other schemes it does not rely on any centralized entity or require specialized trusted hardware at nodes. Therefore, the proposed scheme incurs low overhead and is highly robust.

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

The protocol proposed here is essentially an *incentive-driven* protocol, which ensures that rewards received by intermediate nodes and resource providers for routing and serving requests, respectively, are maximized by following the protocol steps. Please see [2] for a discussion on developing protocols considering the profit-maximizing strategies of individual nodes. A distinguishing feature of the proposed protocol is that it addresses the problem of incentivizing peers for sharing resources and routing messages for others in a unified manner.

II. Related Work

The need for developing protocols for selfish agents (nodes) in P2P systems has often been stressed before (see [2], [3]). The research in [4], [5] provides solution to avoid the 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 deals 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 without any centralized controlling authority, 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 selfishness in routing has been encountered and addressed in the context of mobile ad-hoc networks (see [6], [7]). Some of these proposals can also find application in P2P networks.

III. Network Model

We assume a P2P network model, wherein nodes act selfishly. 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. An example of a transaction is a lookup process, i.e. the process of searching for and downloading a desired data object. Nodes process and forward lookup messages if there is a potential for earning reward in future. The reward can be anything that is deemed to have value, the possession of which adds to a node's utility.

We assume that for each resource there is a single server in the network.[1] Resource indices are replicated at k different nodes, which are called the *terminal* nodes for that resource. They are so called because lookup messages are first routed to these nodes from where they are sent directly to the server node (in one logical hop). Terminal nodes maintain a mapping (called index) from a resource name or ID to the IP address of the server providing that resource. For a resource, say R, its terminal nodes are denoted by T_{R_i} $\forall i \in \{1, \ldots, k\}$. The routing of a message from a client to a terminal node may go through other intermediate nodes. This list of intermediate nodes along with the terminal node (and the client) is referred to as a *request chain*. For simplicity request chains comprising of different terminal nodes for the same resource are assumed to be node disjoint, as shown in Figure 1.

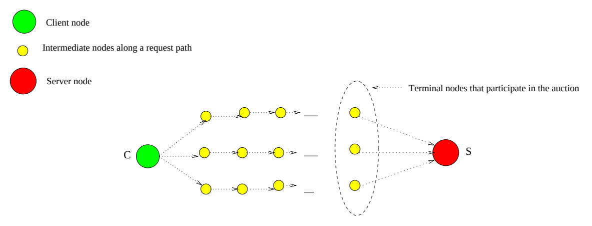
Client node
Intermediate nodes along a request path
Server node
Terminal nodes that participate in the auction
C
S

Fig. 1. Formation of request chains due to the propagation of lookup requests.

For a lookup process initiated by a client, a network can be modelled as comprising of three types of entities - the client itself, the intermediate nodes (including the terminal nodes), and the server providing resource. Nodes incur a cost, due to bandwidth, memory etc., during a lookup process and is represented by MC_x for a node, x. The cost incurred by a server (and also intermediate nodes) increases in proportion to the amount of its traffic and any request offering less than its marginal cost is not fulfilled. Since, clients incur a cost for their lookups, we assume that it is in the clients' best interest to successfully obtain the resource in as few lookup transactions or attempts as possible.

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.[2]

The proposed protocol can be considered as the distributed implementation of a mechanism described in [2]. This is because it is in each intermediate node's best interest to report its true marginal cost for forwarding a lookup request. Moreover, the collection of input values (i.e. marginal costs) and handing out of payments to nodes is done in a distributed manner rather than by some centralized entity.

[1]The terms resource and data are used interchangeably throughout the paper.

[2]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.

IV. Description of Incentive-Driven Protocol

The proposed mechanism correspond to activities in real-world economic markets, where buyers pay money to sellers and intermediaries that facilitate the transactions. However, unlike in real-world, there are no well established protocols (government rules and policies), and institutions and infrastructure (such as stock exchanges) in a typical P2P setting that can govern the parameters (such as the price charged, the place of occurrence etc.) of the transactions. Due to such constraints several non-trivial issues need to be addressed - setting resource prices, determining payoffs to intermediate, preventing cheating etc. We now explain how all such issues are addresses by the proposed protocol. To simplify our discussion, we take an example of a lookup process and see how it is carried out under the given protocol.

A. Parallel Resource Lookup

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. C then sends a separate lookup message towards each of the terminal nodes. Together these parallel lookup messages constitute a single lookup process initiated by C for R.

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 messages. In Section IV-E, we illustrate the significance of $Reqid_{private}$. MC_{total} contains C's marginal cost MC_C. Each intermediate node on receiving the lookup message updates MC_{total} by adding its own marginal cost to the received value.

Intermediate nodes for all the lookup messages route the received lookup message to the next hop neighbor (called the *successor* node) and this process continues till the message reaches the desired terminal node. Since the terminal nodes store the index containing 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 [8], [2]) with all the terminal nodes as the bidders. S provides the resource to the terminal node that offers the highest price. The request chain containing the highest bidder, i.e. the winning terminal node, is called the winning request chain *WRC*.

B. 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 (see [9]).

Since, *S* (the auctioneer) behaves selfishly and tries to maximize its 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, which is described in Section IV-C. 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 fairly among the nodes of the *WRC* (and the client) in proportion to their marginal costs, i.e. nodes with higher marginal costs get a higher proportion of the total profit, and vice versa.

C. Secure Vickrey Auction to Determine the Resource Price

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 message Msg_{cert} to the winning terminal node ($T_{R_{WRC}}$) 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 of all the terminal nodes that participated in the auction (we later explain how this information is utilized by *C* in order to verify the auction results).

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

D. Rewarding Nodes of the WRC

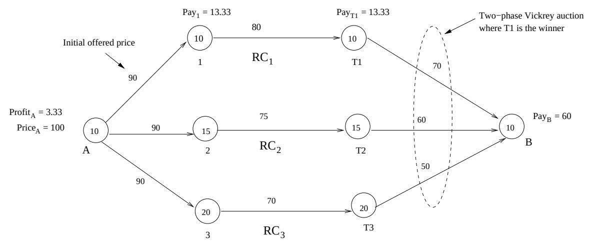

Fig. 2. A lookup example illustrating how payoffs are distributed among the *WRC* nodes based on their marginal costs.

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 determine each *WRC* node's payoff. For example, node *x*'s payoff Pay_x is calculated as follows.

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

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.

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

Let us consider a simple example given in Figure 2 to better understand the above equations. Three request chains (shown as RC_1, RC_2 and RC_3) are formed as part of the lookup process initiated by node *A*. Numbers within the circles represent the nodes' marginal costs. *T1, T2, T3* are the respective terminal nodes that store the resource index, i.e. they store the IP address of node *B* that owns the desired resource. *B* on receiving the lookup requests conducts a Vickrey auction, as a result of which *T1* is selected as the winner, but the price it pays is 60. The results of the auction are sent back to *A*, and also seen by all the intermediate nodes along RC_1. The resulting payoffs to the intermediate nodes and *B* are indicated in the figure. For example, payoff to node *1* is 13.33 (=10 + (10/30)*(70-60)). *A*'s profit share is 3.33 (= (10/30)*(70-60)). Thus, *A* effectively has to pay 86.67(=100-10-3.33) for a resource whose utility to it (after deducting the marginal cost) is in fact 90. Therefore, the proposed scheme based on Vickrey auction ensures that

Msg_{lookup}	$RI_{R_i}, R_i, P_C, MC_{total}, Reqid_{public}, Reqid_{private}$
Msg_{bid}	$E(randKey_i; b_i), Reqid_{public}, MC_{total}$
$Msg_{bid-reply}$	$\cup E(randKey_i; b_i), Reqid_{public}$
Msg_{key}	$randKey_i, Reqid_{public}$
$Msg_{key-reply}$	$\cup randKey_i, Reqid_{public}$
Msg_{cert}	$M_1, M_2, Reqid_{public}, MC_{total}$, IP addresses $\forall T_{R_i}$

TABLE I

VARIOUS MESSAGES COMPRISING THE INCENTIVE-DRIVEN PROTOCOL

everyone, including the client, server, and intermediate nodes constituting the *WRC* benefit, i.e. earn more than their marginal costs, by participating in the lookup process. This potential of earning higher profits motivate nodes to share their resources and forward messages for others.

C after receiving Msg_{cert} determines and takes away its profit share and gives the remainder of its initial offer to the successor node along the *WRC*. The successor node determines its own payoff using Equation 1 and after keeping that amount transfers the remaining to its successor and so on. This process is repeated till the server receives its due payoff. In the above example, *A* after keeping its profit share (and the amount equal to its marginal cost) gives 86.67 to *1*, which after keeping its payoff gives 73.34 to *T1*. Now *T1* after keeping its payoff gives the remaining (i.e., 60 (∼73.34-13.33)) to *B*. The amount received by *B* thus equals M_2 (=60).

A node cannot default on its payment to its successor, since as mentioned earlier, the content of messages (Msg_{lookup} and Msg_{cert}) form a non-refutable contract between a node and its predecessor regarding the amount of money that the node is to receive from its predecessor.

Figure 3 summarizes the steps involved in the incentive-driven protocol. The various messages used, along with the information they contain, are also summarized in Table I for an easy reference.

Step 1: Client initiates the lookup process by sending a lookup message Msg_{lookup} towards $T_{R_i} \forall i \in \{1, \ldots, k\}$
- Intermediate nodes update the value of MC_{total} before forwarding the lookup message
- Lookup messages reach the terminal nodes

/* Vickrey auction - Phase I */
Step 2: Terminal nodes on receiving Msg_{lookup} send Msg_{bid} to the server

Step 3: Server waits for *k* Msg_{bid} messages (i.e. bids) or till some maximum time τ
- Bids are identified as belonging to the same lookup process by using the value $Reqid_{public}$

Step 4: Server sends message $Msg_{bid-reply}$ to the terminal nodes
- After the above step the bidders have encrypted copies of all the bids

/* Vickrey auction - Phase II */
Step 5: Terminal nodes send their secret key to the server in message Msg_{key}

Step 6: Server replies with a message $Msg_{key-reply}$ distributing the secret keys among the bidders

/* Vickrey auction ends */
Step 7: Server sends message Msg_{cert} to $T_{R_{WRC}}$. This message is sent to the client using the reverse lookup path

Step 8: Client verifies the auction results by contacting the bidding terminal nodes

Step 9: Reward is given to the nodes of the *WRC* (including the server and client)

Fig. 3. Incentive-driven protocol steps

E. Threat Models

In this section, we evaluate the robustness of the proposed incentive-driven protocol in the face of nodes' selfishness. In particular, we identify and analyze our protocol for potential threat models and show that truthfully following the protocol steps is the best strategy for the selfish nodes.

1) Threat Model A (Cheating by the auctioneer).: Since the auction by *S* takes place in a completely distributed environment, the bidders are unaware of each others' bids and also cannot monitor *S*'s activities. In such a scenario, using the traditional single-step Vickrey auction, where the bidders directly send their bids in clear to the auctioneer, would enable *S* to easily manipulate the auction results. To understand this, let us again consider the example given in Figure 2. If traditional Vickrey auction is used, then *B* on receiving the three bids of 70, 60, and 50 knows that *T1*, which is the highest bidder, is willing to pay any amount less than or equal to 70 for *R*. Therefore, *B* can send a message to *T1* that it is the highest bidder, but the amount it has to pay (i.e. the second highest bid) is 69. This way *B* wrongly makes an additional profit of 9.

In order to counter the problem of addition of fake bids (for example, the bid value 69 as explained above) and manipulation of submitted bids by an auctioneer, we use a two-phase Vickrey auction as described in Section IV-C. Now the auctioneer, before it can read the bids, has to give encrypted copies of all the received bids back to the bidders. Therefore, in the above example, *B* is unable to send fake bid 69 after finding that *T1*'s bid is 70.

One might argue that there is a possibility for the auctioneer to send different encrypted bids to different bidders if it stands to gain by doing so. However, this strategy would not be effective unless the auctioneer has prior information about the bids it is going to receive, as shown next. Using the same example, let *T1*'s bid be 69, instead of 70 as anticipated by *B*. Therefore, during the first phase of the secure Vickrey auction protocol, *B* encrypts values 50, 60 and 70 and sends it to all the bidders. Here 50 and 60 represent the actual bids and 70 is the fake bid. In the second phase, *T1* (and also *T2* and *T3*) after receiving the decryption keys finds that the highest bid is 70 and that it has lost the auction. Thus, none of the bidders get selected as the winner and *B* by faking the bids gets a payoff of 0 as opposed to 60 that it could have received by not cheating.

It is possible that even the knowledge of the distribution of the bid values and the number of bids might be exploited by the auctioneer. The auctioneer can send different combinations of encrypted bids to different bidders in order to fake the auction results and thus maximize its expected profit. Such situations are easily handled by the solution proposed for the next threat model. Basically, the strategy is to ensure that the auctioneer is unable to send different bids to different bidders. This is achieved without requiring any costly and difficult to implement communication among the bidders themselves.

2) Threat Model B (Collusion between S and $T_{R_{WRC}}$).: The proposed protocol relies on the fact that correct auction results are sent back to *C*, so that the reward is fairly distributed among all the nodes comprising the *WRC*. However, it is possible for *S* and $T_{R_{WRC}}$ to collude and make higher profits by including a fake second highest bid value in Msg_{cert}. For example, in Figure 2, by including the value of M_2 as 69 (instead of 60) in Msg_{cert}, *T1* receives the payoff of 79.34 from node *1*, instead of 73.34 that it receives by not colluding. This higher payoff can be shared between *S* and $T_{R_{WRC}}$ and so they both benefit with this collusion.

As mentioned earlier, the message Msg_{cert} sent back to *C* includes the information (i.e. the IP addresses) of all the terminal nodes that participated in the auction. *C* on receiving this information can verify the truthfulness of the received auction results by contacting any (or all) of the listed terminal nodes. These terminal nodes are given incentive to reveal the truth, i.e. disclose the true values of the highest and second highest bid in the auction. The terminal node that identifies that there is a discrepancy (if any) between the auction results received by *C* and the actual values, is referred to as the whistle blower $T_{R_{WB}} \exists WB \in \{1, \ldots, k\}$. *C* can give $T_{R_{WB}}$ part of the money that it saves by detecting the collusion.[3]

A lookup transaction can be considered as a one-shot game in which each participant tries to maximize its profit in a single play of the game. One-shot model is reasonable to assume because the network under consideration is large, distributed, and dynamic. Moreover, it is difficult for nodes to monitor and keep track of others that do not fulfill their collusion agreement. Thus, the terminal nodes have incentive to become a whistle-blower, as they get additional reward from the client (possibly in addition to what they receive from the server for not revealing the truth).

Moreover, *C* upon contacting the terminal nodes ensures that they have the same auction results, i.e. they received the same encrypted bids from the auctioneer during phase one of the auction. Thus, any cheating by the auctioneer, such as sending different encrypted bids to different bidders, as mentioned in threat model A, can be easily detected by *C*. This is achieved without incurring excessive message communication overhead required in any bidder discovery and verification protocol, in which bidders identify each other and cross-check each others' bid values.

In effect, *C* acts as a centralized controller for its lookup process and ensures that no cheating by the auctioneer and/or collusion between the auctioneer and winning terminal node takes place.[4]

3) Threat Model C (Sending incorrect terminal nodes information).: The prevention of collusion in threat model B relies on the fact that the information about the terminal nodes sent by *S* back to *C* in Msg_{cert} is correct. However, it is possible for *S* to include fake information about nodes, which it control or with whom it has prior collusive agreement, such that they are guaranteed not to be the whistle blowers. *C* will then have no way of cross-checking the bid values and would end up paying more than what it should. To prevent such a possibility, *C* includes a unique request ID $Reqid_{private}$ in each of the lookup request messages it sends. Upon contacting a terminal node, *C* requests the $Reqid_{private}$ value that the node has to make sure that the value is indeed one of the values it initially included in a lookup message. This provides a method for terminal nodes' authentication, as *C* can be sure that it is interacting with a valid terminal node.

4) Threat Model D (Over-reporting of marginal costs and under-reporting of utility values).: We have shown how Vickrey auction can be used to establish utility-driven pricing in a completely untrusted and distributed P2P environments. Vickrey auction results in fair pricing, in the sense that it reward clients for being truthful in stating their true utilities for the resources, and also the intermediate nodes for revealing their true marginal costs for forwarding the lookup requests.

An increase in the MC_{total} value for a request chain lowers its final bid, thereby reducing its chances of winning the auction. If intermediate nodes run specialized learning algorithm and gather privilege information about the network state, such as other intermediate nodes' marginal costs that comprise the different request chains, then they may benefit (i.e. make higher profits) by quoting a higher marginal cost and still be part of a *WRC*. Such information, however, is not easy to obtain in a highly dynamic environments, and also the information about current network state may not remain valid even in near future periods. In addition, implementing such algorithms can be expensive. Therefore, to minimize MC_{total} it is in each node's best interest to report its true marginal cost.

One may argue that an intermediate node can increase its profit by only slightly increasing its true marginal cost, while not jeopardizing its chances of still being part of a *WRC*. However, we show that in the absence of any privilege information, revealing true marginal cost is the optimal strategy for a node. This result is summarized in the form of the following lemma.

Lemma 1: Given that a client's utility for a resource being looked up is bounded (for example, in Chord [10] if it is less than four times the total marginal cost of nodes comprising the *WRC*), the best strategy for an intermediate node is to report its true marginal cost while forwarding a lookup request.

Proof: Let the price offered by bidders (terminal nodes) to an auctioneer (server) are in the interval from $[0, P]$. There are k ($k >> 1$) bidders and we assume that all the k bids are uniformly distributed in the interval $[0, P]$. The average distance

[3]Note that the terminal nodes have verifiable copies of the encrypted bids and corresponding keys that they receive from the auctioneer. This verification is possible due to message non-repudiation.

[4]In fact, the proposed solution is effective even if all the nodes (except the client) of the *WRC* collude to get a higher payoff for themselves. This is because the client's profit share is dependent only on its own marginal cost as well as the auction results, which it can verify from the terminal nodes.

between a bid and the next higher bid (assuming that there is one) is $\frac{P}{k+1}$.

Further consider a node, i, which is part of some request chain and has a true marginal cost of MC_i. For simplicity, we assume that all the nodes belonging to *i*'s request chain have the same marginal cost. We would show that if all the other nodes (except *i*) that are part of the lookup process report their true marginal costs, then the best strategy for node *i* is to report its true marginal cost only.[5] To understand this, let node *i* falsely increase its marginal cost by y ($y << MC_i$) and report it as $MC_i + y$ instead of MC_i. Some other variables that we use are defined below.

T = total marginal cost of the nodes belonging to the same request chain as node *i*. T includes MC_i.

τ = price offered by the terminal node of node *i*'s request chain to the server. $\tau \in [0, P]$.

For simplicity, we say that $T + \tau = P$. This is based on the assumption that P represents the client's utility for the resource being looked for and that the client uses its true utility value while initiating the lookup process.

Since node i gets a payoff only if the terminal node of its request chain wins the auction, its payoff when it acts truthfully and falsely, represented as E_T and E_F, respectively, are calculated as follows:

$$E_T = (\frac{\tau}{P})^{k-1} * [\frac{P}{k+1} * \frac{MC_i}{T} + MC_i] \qquad (3)$$

$$E_F = (\frac{\tau - y}{P})^{k-1} * [(\frac{P}{k+1} - y) * \frac{MC_i + y}{T + y} + MC_i + y] \quad (4)$$

The first term on the right hand side of both the Equations 3 and 4 describe the probability that node *i* is part of the *WRC* and the second term gives the payoff that it subsequently receives. We now proceed to show that under the condition when $P < 4 * T$, E_F is less than E_T. For tractability, we further assume a Chord based P2P network, where the number of neighbors of a node is approximately twice the average hop length of a lookup path. In other words, $T = \frac{k}{2} * MC_i$.

E_T is greater than E_F if $E_F - E_T < 0$, i.e. if

$$\frac{\tau^{k-1}}{P^{k-1}} * (1 - \frac{y}{\tau})^{k-1} * [(\frac{P}{k+1} - y) * \frac{MC_i + y}{T + y} + MC_i + y] - (\frac{\tau}{P})^{k-1} * [\frac{P}{k+1} * \frac{MC_i}{T} + MC_i] < 0$$

or

$$\frac{\tau^{k-1}}{P^{k-1}} * (1 - (k-1) * \frac{y}{\tau}) * [(\frac{P}{k+1} - y) * \frac{MC_i + y}{T + y} + MC_i + y] - (\frac{\tau}{P})^{k-1} * [\frac{P}{k+1} * \frac{MC_i}{T} + MC_i] < 0$$

In the above, we use binomial expansion to solve $(1 + \frac{y}{\tau})^{k-1} = 1 - (k-1) * \frac{y}{\tau}$. Higher order terms are ignored, since they anyway further reduce E_F. Solving the above inequality we get,

[5]From Equation 1 we know that a node can get a higher payoff by falsely reporting a higher marginal cost value.

$$\frac{\tau}{2T} + \frac{k * y}{\tau} < \frac{T}{\tau} + \frac{3}{2} \Rightarrow \frac{\tau}{2T} < \frac{3}{2}$$

Thus, if $\tau < 3T$, we have $E_F < E_T$. This proves that if the client's utility (or P) is less than $4T$, truthfully reporting the marginal cost maximizes one's (here node *i*'s) payoff. ■

Moreover, a client's goal to obtain a resource in a single lookup transaction is best served if it sets maximum possible price for the desired resource and that price is the client's utility for the resource. The server's marginal cost of serving the request (and of the intermediate nodes) is unknown to the client and can be high depending on the number of requests it is currently serving. Together these factors ensure that using the actual utility value for setting the offered price is the best strategy for a client.

V. Conclusion and Future Work

In this paper we have presented an incentive-driven protocol for searching and trading resources in uncooperative P2P networks. The protocol provides incentive to nodes to share their resources and route messages for others. We developed resource pricing in a dynamic and selfish environment and how to avoid collusion among nodes. Our proposed protocol takes selfish behavior of network nodes into account and ensures that their rewards are maximized if they adhere to the protocol steps.

References

[1] A. Eytan, and A. H. Bernardo. Free Riding on Gnutella. First Monday, vol. 5, No. 10, Oct. 2000.

[2] 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.

[3] D. Geels, and J. Kubiatowicz. Replica Management Should Be A Game. *In Proceedings of the SIGOPS European Workshop*, 2002.

[4] 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.

[5] 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.

[6] 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.

[7] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad-hoc nertworks. *In Proc. of ACM Journal for Mobile Networks (MONET), special issue on Mobile Ad Hoc Networks*, summer 2002.

[8] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance, pages 8-37*, 1961.

[9] 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.

[10] 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.