An Incentive Driven Lookup Protocol For Chord-Based Peer-to-Peer (P2P) Networks

Rohit Gupta and Arun K. Somani
Dependable Computing and Networking Laboratory
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 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 provides efficient and natural means to prevent free-riding problem in Chord and does not require prior trust relationships among nodes. Therefore, it incurs low overhead and is highly robust. In addition, we evaluate the performance of Chord [1] for providing routing in a network of selfish nodes and prove that it has good structural properties to be used in uncooperative P2P networks.

I. 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-based P2P network model and describe a novel strategy for carrying out lookups in such networks. Our proposed scheme provides an efficient and natural means to prevent free-riding problem in Chord without requiring any prior trust relationship 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 the reward received by the 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 the fact that whether they themselves are currently overloaded or not, for example. Please see [3] 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 to share their resources and route messages for others in a unified manner.

We evaluate the performance of Chord for providing routing services 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 II is on related work, Section III describes the network model. Section IV gives a detailed description of the proposed lookup protocol including the various possible threat models it is designed to withstand. Section V presents a resource index replication strategy that is useful in dealing with selfish nodes in Chord. Section VI explains why Chord is a good protocol to be used in a network with selfish nods. We conclude in Section VII.

II. Related Work

The need for developing protocols for selfish agents (nodes) in P2P systems has often been stressed before (see [3], [4], [5]). The research in ([6], [7], [8], [9]) 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 with others 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 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 mechanism proposed in [10] requires a node to route messages for others in order to obtain routing service in return. The authors in [10] develop a trust and security architecture for a routing and node location service that uses a trust protocol, which describes how honest nodes should perform. The protocol proposed in this paper not only provides incentive to nodes to route messages for each other, but also avoids the problem of free-riding associated with other network resources, such as

data.

The problem of selfishness in routing has been encountered and addressed in the context of mobile ad-hoc networks (see [11], [12]). Some of these proposals can also find application in P2P networks. Below we describe some of the recent research effort that promotes cooperation among selfish mobile ad-hoc network nodes.

In [12], Buttyan and Hubaux proposed a scheme based on credit counter. In it nodes pass each packet to its security module. The security module maintains a counter, called *nuglet counter*, which is decreased when the node wants to send a packet as originator, and increased when the node forwards a packet. The nuglet counter is protected from illegitimate manipulation by the tamper resistance of the security module. Another approach is exemplified by the Sprite system [11], which provide incentive to mobile nodes to cooperate. When a node sends its own messages, the node looses its credit (or virtual money) to the network because other nodes incur a cost to forward the messages. On the other hand, when a node forwards others' messages it gains credit. The system determines payments and charges from a game-theoretic perspective and is effective in motivating nodes to behave honestly, even when a collection of the selfish nodes collude. However, Sprite relies on a centralized trusted third-party to achieve its goals.

III. Network Model

A. Chord Overview

Chord [1] supports just one operation, i.e. given a key, it returns the node responsible for that key. Each Chord node has a unique *m*-bit identifier (Chord ID), obtained by say, hashing the node's IP address. Chord views the IDs as occupying a circular identifier space. Keys are also mapped into this ID space, by hashing them to *m*-bit key IDs. Chord defines the node responsible for a key to be the *successor* of that key's ID. The *successor* of an ID *j* is the node with the smallest ID that is greater than or equal to *j* (with wrap-around).

Every Chord node maintains a list of the identities and IP addresses of its *r* immediate successors on the Chord ring. The fact that every node knows its own successor means that a node can always process a lookup correctly: if the desired key is between the node and its successor, the latter node is the key's successor; otherwise the lookup can be forwarded to the successor, which moves the lookup strictly

closer to its destination. In a system with *N* nodes, lookups performed only with successor lists require an average of $N/2$ message exchanges. To reduce the number of messages required to $O(\log N)$, each node maintains a finger table with *m* entries. The i^{th} entry in the table at node *j* contains the identity of the first node that succeeds *j* by at least 2^{i-1} on the ID circle. A new node initializes its finger table by querying an existing node.

B. Model Details

The model of network assumed here is a Chord-based P2P network. The nodes in the network 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. For now, we assume that electronic money [13], [14] is being used as reward and there is a PKI-based [16] security infrastructure, such that each node has a unique public-private key pair.

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 in the network, 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} $\forall i \in \{1, \ldots, k\}$, 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 ring. The method by which these mappings are determined is explained in Section V.

The client *C* gains utility U_R^C by obtaining R and thus can offer a maximum price equal to U_R^C in order to obtain the resource. Since the client also incurs a cost for each lookup that it initiates, we assume

that it is in the client's best interest to successfully obtain the resource in as few lookup transactions or attempts as possible.

The message communication is reliable, i.e. a message sent is received by the intended receiver in bounded time without any distortion. Moreover, 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]

IV. 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.

A. Parallel Lookup Towards the Terminal Nodes

The client *C* before initiating the lookup process estimates its utility of the resource *R* (U_R^C) 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*, including any intermediate node, on receiving multiple lookup messages know that they all 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 $Reqid_{private}$

[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.

is included in each of the lookup message. In Section IV-E, 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 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 messages route the received lookup message to the next hop neighbor and this process continues till the message reaches the desired terminal node. 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 (Vickrey auction [17], [3]) with all the terminal nodes as the bidders. *S* provides the resource to the terminal node that offers it the highest price.

B. Bidding for 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 [18].

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.

[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. The "successor" here refers to the next hop neighbor along the lookup path. From now onwards, mostly it is used to refer to the next hop neighbor only and we prefix it with "Chord", i.e. "Chord successor", whenever using it as described in the Chord protocol. Also, the term "predecessor" refers to a previous hop node along the lookup path.

- 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. Phase one of the protocol is similar to an earlier protocol in [19] for secure second-price auctions. In both the protocols, bidders initially send encrypted copies of their bids to the auctioneer.

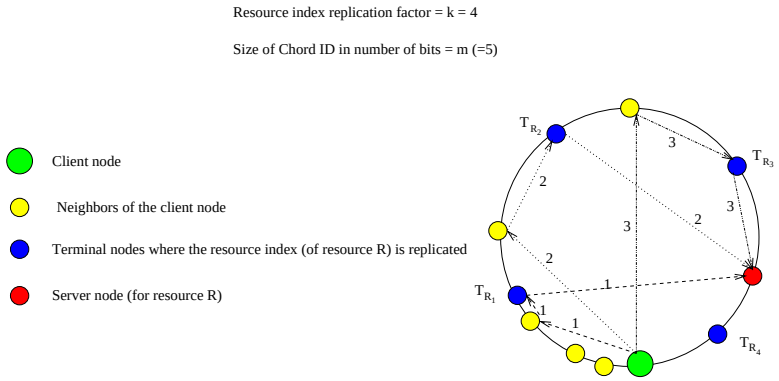

Fig. 1. Lookup message propagation in the *incentive driven lookup* protocol.

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 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 Figure 1, the client *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 reached during the lookup process may be a subset of those containing the resource index. However, for simplicity, it is assumed that all the terminal nodes for a resource are contacted and participate in the lookup process.

Figure 2 depicts an equivalent representation of Figure 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

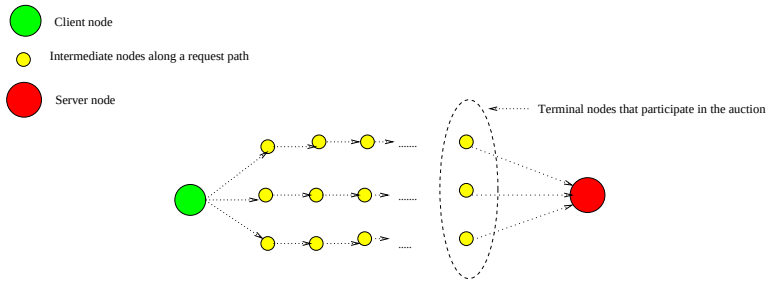

Fig. 2. Formation of request chains due to the propagation of lookup requests.

profit is shared fairly 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.

C. Secure Vickrey Auction to Determine Resource Prices

The server *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 of their bids $E(randKey_i; b_i)$ in message Msg_{bid} to *S*. Here b_i is the bid value sent by terminal node T_{R_i} using a randomly chosen secret key, $randKey_i$. Msg_{bid} also include the value $Reqid_{public}$, so that *S* determine that the bids pertain to the same lookup process.[3] 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, a bidder after receiving $Msg_{bid-reply}$, send its secret key in message Msg_{key} to server *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 $T_{R_{WRC}}$) 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}),

[3]It must be noted that only one of the terminal nodes is given the resource, since the client does not make multiple payments upon receiving the same resource from different nodes.

and the IP addresses and Chord IDs of all the terminal nodes that participated in the auction (we later explain how this information is utilized by *C* 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*.[4] The knowledge of auction results also enable *C* to determine the price it finally has to pay for R. The calculation of exact payoff values are discussed below.

D. Rewarding Nodes of the WRC

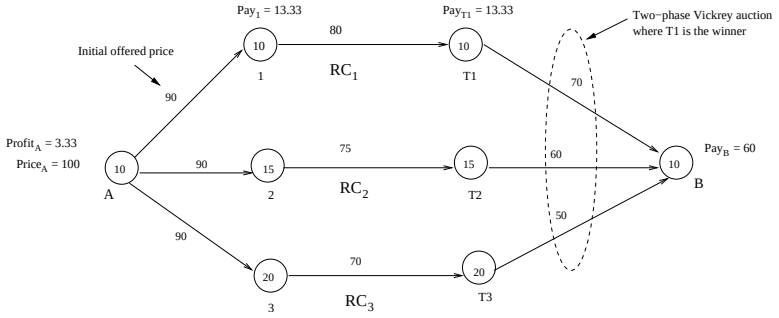

Fig. 3. A lookup example illustrating how payoffs are distributed among the *WRC* nodes based on their marginal costs.

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 3 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

[4]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).

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}	$M_1, M_2, Reqid_{public}, MC_{total}$, Chord ID and IP addresses $\forall T_{R_i}$

TABLE I

VARIOUS MESSAGES COMPRISING THE INCENTIVE DRIVEN LOOKUP PROTOCOL

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 also 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 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.

In the above 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 4 summarizes the steps involved in the incentive driven lookup protocol. The various messages used, along with the information they contain, are also summarized in Table I for an easy reference.

E. Threat Models

In this section, we evaluate the robustness of the proposed incentive driven lookup 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.

Step 1: Client initiates the lookup process by sending a lookup message Msg_{lookup} towards $T_{R_i} \forall i \in \{1, \ldots, k\}$
- Routing of lookup messages is based on the Chord routing protocol
- 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 from which the bid was received
- 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. 4. Incentive driven lookup protocol steps

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 3. 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 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 (as in Byzantine general's problem [20]) 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. Moreover, 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.

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 3, 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 and Chord IDs) 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.[5]

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

[5]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 the message non-repudiation mechanism described in Section III-B.

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.

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).: 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 the *WRC*. Such information, however, is not easy to obtain in a highly dynamic environments and also the information about the

current network state may not remain valid at all even in near future periods. Thus, in the absence of any privilege information, revealing true marginal costs is the optimal strategy for nodes. In Appendix VIII-A, we show that a node's expected payoff is lower if it falsely increases its *MC* even by a small value.

F. Protocol Overhead

We must admit that our incentive based lookup protocol designed to address nodes' selfishness adds some overhead to the system. The overhead is primarily due to two reasons - message communication involved in formation of request chains including validation of auction results by the client and the computation involved in message encryption and decryption to achieve message non-repudiation. However, message non-repudiation is not needed when the client contact the terminal nodes for validating the auction results.

The maximum message processing overhead is incurred by the client but it does not increase linearly with the system size and is equal to *O(logN)* for a *N*-node system. The number of messages processed by an intermediate node is *O(1)* and that by the server equals *O(k)*. The maximum number of nodes involved in the lookup process are *O(klogN)*, where *k* is the number of request chains and *O(log N)* is the length of each request chain. As can be seen the number of messages exchanged is still significantly lower than that required when flooding the network for searching an object. The proposed protocol deals with the selfishness problem of nodes without relying on any centralized entity or deploying specialized trusted hardware in each network node.

Our protocol relies on giving incentives to nodes in order to achieve cooperation in message routing and resource sharing. The incentives are typically some form of reward or money that require an electronic payment infrastructure. We are currently developing a framework for reputation management that allows using reputation as a form of currency and it thus obviates the need for any payment infrastructure.

V. Resource Index Replication

The proposed 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 (i.e. more disjoint the request chains are), higher is the robustness of the pricing scheme. If normal Chord index replication is used, i.e. storing the 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 payoff 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 ring 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 disjoint from each other 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),[6] then the *k* resource index replicas map to the following Chord IDs.

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

The index values are then stored at the Chord successors (the terminal nodes) of the IDs represented by $RI_{R_i} \forall i \in \{1, \ldots, k\}$. 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. Since in the Chord routing protocol, a lookup request makes smaller and smaller hops as it nears its destination, a significant percentage of the accesses to the *k* index replicas always pass through a small subset of nodes, thereby increasing the possibility of collusion among them. On the other hand, uniformly distributing 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 Figure 5, 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.

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

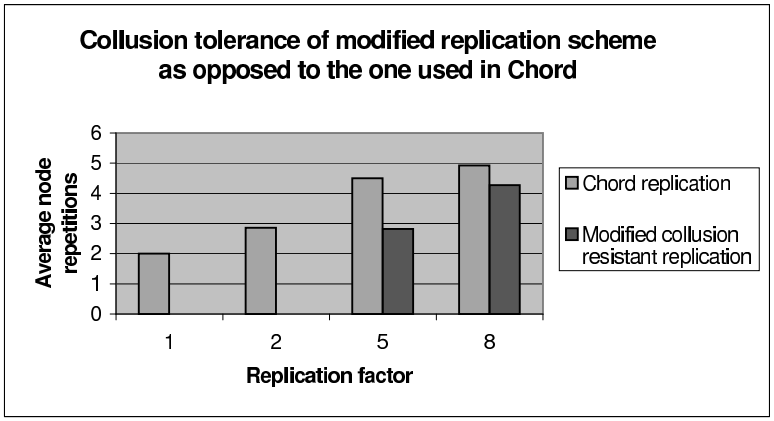

Fig. 5. Average number of repetitions of intermediate nodes that appear in multiple lookup paths to the resource index replica copies. Size of the network $N = 500$

VI. Selfish Network Topology

So far we have assumed that nodes form a Chord ring and the lookup messages are forwarded in accordance with the Chord routing protocol. Now, we investigate how relevant is the assumption that nodes would truthfully follow the Chord protocol and how else connectivity among nodes may be achieved. Since the nodes are selfish and join the network to obtain resources and maximize their profits, the manner in which they choose their 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* (which is at least equal to their marginal costs) before forwarding a lookup request to their next hop neighbors. Thus, fewer intermediate nodes generally translate to higher profits for the client. Therefore, closer the client is to the server, the lower is the price that it has to pay for the resource.

A node can make higher profits by being close to as many different resources that it requires, as possible. However, if the location of those resources 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 as described in the previous section. Consider the network shown in Figure 6, if a new node *1* joins the network and has to fill up m (= *5*) entries in its routing table then it can select neighbors as per the greedy routing approach instead. Node *1* tries to minimize the number of hops by being greedy, whereas other nodes in the network continue to follow the Chord

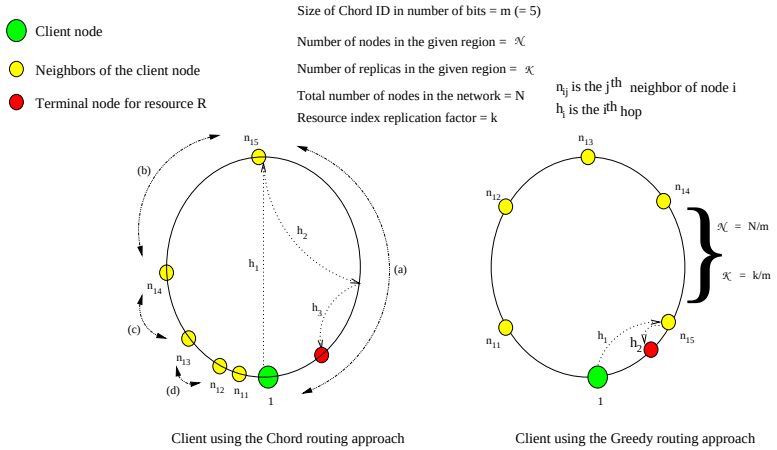

Fig. 6. Comparison of Chord and the greedy routing approach. Network nodes are uniformly distributed in the Chord ID space.

protocol. For example, if node *1* needs to send a message to the terminal node for resource R, it can do it in two hops using the greedy strategy as opposed to three hops required using Chord.

In Figure 6, 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) that required by the Chord protocol.[7] Even for the other regions of Figure 6, the greedy approach appears to perform comparably to Chord. This is because node *1* 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.

Thus, we see that nodes possibly have motivation to not follow the Chord protocol and instead utilize the fact that other nodes follow Chord, in order to make higher profits for themselves. If everyone in the network knows this fact, then they too would follow the greedy approach. But if this happens the whole routing system would break down, in the sense that instead of $O(logN)$ routing provided by Chord, $O(N/k)$ hops would be required due to the resulting sequential search for the resource indices. The minimization of the number of routing hops by the greedy approach when everyone else follows Chord, however, is not always correct. We prove in Appendix VIII-B that in a large network, on average the performance of greedy routing approach is no better than Chord. The following lemma follows directly

[7]This assumes that *N* is large and the nodes are uniformly distributed around the Chord network.

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.

VII. 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 and ensures that their rewards are maximized if they adhere to the protocol steps. We used Vickrey auction for setting resource prices in P2P networks and described how it can implemented in a completely distributed and untrusted environment. In the process, we address the known problems in Vickrey auctions, that of a selfish auctioneer and provided a solution to deal with it.

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 follow 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] J. Shneidman, and D. C. Parkes. Rationality and Self-Interest in Peer to Peer Networks. *In Proceedings of IPTPS 2003, Berkeley*, February 2003.

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

[6] H. T. Kung, and W. Chun-Hsin. Differentiated Admission For Peer-To-Peer Systems: Incentivizing Peers To Contribute Their Resources. *In Proceedings of the 2003 Workshop on Economics of Peer-to-Peer Systems, Berkeley CA*, 2003.

[7] 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.

[8] T. J. Ngan, D. S. Wallach, and P. Druschel. Enforcing Fair Sharing of Peer-to-Peer Resources. *In Proceedings of IPTPS 2003, Berkeley*, February 2003.

[9] 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.

[10] T. Moreton, and A. Twigg. Enforcing collaboration in P2P routing services. *In First International Conference on Trust Management*, 2003.

[11] 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.

[12] 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.

[13] M. Bellare, J. Garay, C. Jutla, and M. Yung. VarietyCash: a multi-purpose electronic payment system. *In Proc. Of 3rd Usenix Workshop on Electronic Commerce, pages 9-24*, August 1998.

[14] G. Medvinsky. A Framework for Electronic Currency. *PhD thesis, USC*, 1997.

[15] N. Asokan. Fairness in electronic commerce. *PhD thesis, University of Waterloo, Ontario, Canada*, May 1998.

[16] C. Pfleeger. *Securiy in Computing*. 2nd edition, Prentice Hall, 1997.

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

[18] 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.
[19] B. Felix. Cryptographic Protocols for Secure Second-Price Auctions. In M. Klusch and F. Zambonelli, editors, Cooperative Information Agents V, volume 2182 of Lecture Notes in Artificial Intelligence, pages" 154-165, Berlin et al., 2001.
[20] M. Pease, R.Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal of ACM, 27 (180), pp.228-234.*

VIII. APPENDIX

A.

Lemma 2: Given that a client's utility for a resource being looked up is bounded (for example, 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 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.[8] 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:

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

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

$$E_F = (\frac{\tau - y}{P})^{k-1} * [(\frac{P}{k+1} - y) * \frac{MC_i + y}{T + y} + MC_i + y] \tag{5}$$

The first term on the right hand side of both the Equations 4 and 5 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. In Chord, 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,

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

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. ■

B.

Theorem 1: In a large network, on average the greedy approach requires the same number of hops as that required by the Chord routing approach.

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

In Chord, the average number of hops required to reach any node from (say) *1* is $1/2 * (logN)$. Therefore, in Figure 6, 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 approach to reach one of the k terminal nodes: The neighbors (i.e. the entries in the routing table) 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, the client 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.

$$(1 + \frac{1}{2} * log(\frac{N}{k})) \tag{6}$$

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 the client (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)[9], and so on (up to *m* such terms).

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

$$\frac{(1 + \frac{1}{2} * log(\frac{N}{k})) + (1 + \frac{1}{2} * log(\frac{N}{k})) + \ldots}{m} \tag{7}$$

Simplifying 7 we get,

$$(1 + \frac{1}{2} * log(\frac{N}{k})), \tag{8}$$

which is same as the number of hops given by Equation 6.

[9] $(1 + \frac{1}{2} * log(\frac{N/4}{k/4}))$.

The results from our simulations (see Figure 7) also confirm the fact that nodes cannot benefit by selecting the greedy routing strategy.

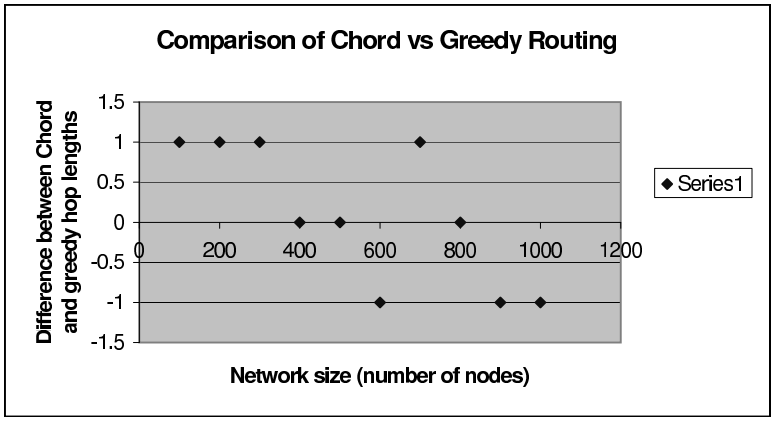

Fig. 7. Comparison of Chord and the greedy routing approaches with regards to the hop-length.

Figure 7 indicates the difference in the observed number of hops when greedy routing strategy is used as opposed to normal Chord routing. We did simulations for varying number of total nodes and averaged the number 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. ■