Utilizing Node's Selfishness for Providing Complete Anonymity in Peer-to-Peer Based Grids

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

Abstract

In this paper, a novel protocol for providing both client and server anonymity in peer-to-peer (P2P) based grids is presented. The protocol assumes individual nodes or users to be utility maximizing agents, and relies on an auction mechanism for trading of resources among them. The resources here can refer to data files, storage capacity, or computation power (i.e., CPU cycles) etc. The protocol is inherently anonymous, light-weight, and incentive-compatible. Incentive compatibility implies that the protocol takes into account the selfishness of users; as would be seen the utilities of users are maximized by truthfully following the protocol steps. Moreover, if the protocol is truthfully followed by the users, anonymity to both the clients and servers of all the transactions are guaranteed. Furthermore, unlike other schemes, the proposed protocol does not rely on any trusted centralized entity or require specialized encryptions to be performed by the users. Thus, the protocol incurs very low overhead on the system and is light-weight. In brief, the proposed protocol provides efficient and natural means to provide anonymity in P2P based grids, and is easily deployable in a large un-trusted Internet-scale setting.

Index Terms

Anonymity, peer-to-peer, selfishness, incentives

I. Introduction

Peer-to-peer (P2P) networks [18], [8], [9], [24] are flexible distributed systems that allow nodes (also called peers) to act as both clients and servers and provide services to each other. P2P is a powerful emerging networking paradigm that permits sharing of virtually unlimited data and computational resources in a completely distributed, fault-tolerant, scalable, and flexible manner.

Much of the focus on grid computing [10] to date has assumed the existence of static grid sites, which have out-of-band trust relationship among themselves. However, there is also a growing body of research that allows small workgroups and laboratories to develop a less formal and organized grid environment. Such an ad-hoc computing environment eliminates the need for grid infrastructure administration and instead offer a more peer-to-peer type grid environment to users (for more details on ad-hoc grid computing see [10]). In the near future, with the tremendous growth and development in P2P and grid computing it is possible to have a confluence of the two technologies, whereby large grids are created and organized in a P2P fashion. It is these P2P based grids that are the focus of this paper.

The goal of this work is to provide complete anonymity to the users of a P2P system, i.e., if a user initiates a request, say for data download or remote task execution, it should remain anonymous; likewise a user serving the request should not be traceable. For instance, in a commercial application, enterprises may be accessing resources residing at third-party computing facilities, e.g. delegating a computation-intensive job or acquiring a large amount of data. Since the third party could have significant findings about an enterprise activity if the enterprise identity is disclosed, anonymous communication in such scenarios is as important as other security issues.

Anonymity refers to the state that an entity is not identified in the communications with others. As discussed in [2], anonymous communication may have one or more of the following properties: sender anonymity, receiver anonymity, and unlinkability. Sender anonymity means that when a message is observed, the sender cannot be identified; receiver anonymity means that the receiver cannot be identified. Unlinkability means that the relationship between the sender and the receiver in the communication cannot be identified, even if sender anonymity or receiver anonymity cannot be guaranteed. Some anonymity mechanism may provide anonymity against one type of threat but not against another type. For example, using a proxy between senders and receivers may provide sender anonymity against the receiver and vice versa, but may not provide any anonymity against an eavesdropper who can observe all messages from and to the proxy.

In this paper, a novel protocol for providing both client and server anonymity (and hence also unlinkability) in P2P networks is presented.[1] The protocol assumes individual nodes or users to be utility maximizing agents, and relies on an auction mechanism for trading of resources among them. The resources here can refer to data files, storage capacity, or computation power (i.e., CPU cycles) etc. The protocol is inherently anonymous, light-weight, and incentive-compatible. Incentive compatibility implies that the protocol takes into account the selfishness of users .[2] The authors in [1] also propose the idea of using economic incentives for building decentralized anonymity infrastructure and motivating users to participate in such a system. Moreover, if the protocol is truthfully followed by the users, anonymity to both the clients and servers of all the transactions are guaranteed. Furthermore, unlike other schemes, the proposed protocol does not rely on any trusted centralized entity or require specialized encryptions to be performed by the users. Thus, the protocol incurs very low overhead on the system and is light-weight.

The authors believe that the proposed scheme is easily deployable in a large untrusted Internet-scale setting. To the best of the author's knowledge this is the first work that uses an incentive strategy to address the problem of both client and server anonymity in a single unified protocol.

The paper is structured as follows. Section VI is on related work. Section II describes the system model. Section III gives a detailed description of the proposed protocol. The anonymity provided by the protocol is studied in Section IV. The overhead incurred by the protocol is summarized in Section V, and finally the paper conludes in Section VII.

II. System Model

A P2P network substrate that is used for network connectivity and resource lookups has been assumed. It must be noted that in a P2P based system there are no well-defined servers, and nodes act as both servers and clients at different times. Nodes organize themselves into a logical Chord ring and follow the Chord lookup protocol for resource discovery. A brief description of Chord has been provided in appendix 1. It is also assumed that a node's Chord identity (or simply Chord ID) is derived by applying an appropriate hash function to its IP address, and thus remains fixed for the node.

Message communication is reliable, i.e., a message sent is received by the intended receiver in bounded time without any distortion. Unless otherwise specified, all message communication is assumed to provide message non-repudiation. The protocol relies on message non-repudiation to ensure that nodes do not go back on their *commitment* as suggested by the content of the

[1]Here clients and servers are nodes in a P2P system. A node initiating a request for a service (or resource) is referred to as the client, and a node that serves the request is referred to as the server (for that particular request).

[2]Please see [19] for a discussion on developing protocols considering the profit-maximizing strategies of individual nodes. In fact, the protocol proposed here is in line with the idea of using mechanism design for solving distributed computing problems as described in [19]

messages sent by them. There exist a mechanism to punish nodes if it can be proven that they do not fulfill their commitments.[3]

Nodes are autonomous rational agents in a game-theoretic sense. By autonomous it is meant that nodes are completely free to choose their actions. The profit from a transaction is equal to the difference between the *reward* that a node earns and the *cost* that it incurs by participating in the transaction. The goal of rational agents is to maximize their utilities (or profits) during each network transaction. Nodes participate in a transaction if there is potential for making a profit.

The reward can be anything that is deemed to have value, the possession of which adds to a node's utility. For simplicity, it is assumed that rewards are electronically processed and a secure payment mechanism among peers is in place. One such secure distributed peer-to-peer payment mechanism, called Karma [28] is described in appendix 2. 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 term "marginal" reflects the fact that this cost value is for a given transaction and represents the additional work that a node has to do for participating in the transaction. The cost incurred by a node increases in proportion to the amount of traffic it is currently handling, and any request offering less than its current marginal cost is not fulfilled.

A *transaction* is an event wherein some peer requests a service from other peers in the network. The definition of a *service* can include anything from sharing data and compute power to routing messages etc. For example, for *music file sharing*, the lookup process initiated by a peer to download some music file corresponds to a transaction. The lookup process includes both searching for and obtaining the file. Similarly, for *compute power sharing*, a transaction corresponds to searching for an appropriate computing node and then sending it a task for execution. In the remainder of the paper, the term service is used to refer to data file sharing, however, the proposed protocol is general enough to be applicable for other services also.

In each transaction resources are traded using an auction protocol; Vickrey auction [29], [19] is used in which the highest bidder wins the auction, but the price that it pays 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.

Some nodes in the network can be malicious, whose goal is to compromise the identity of the nodes that request and/or provide a service. Malicious nodes do not aim to maximize their profits (and in fact are prepared to incur loss), and can work in collusion, so as to identify who originated a request and who served it. An *internal and static adversary* model [12] is assumed, where a) the adversary can observe lookup requests at compromised nodes but cannot observe the links (internal) and b) the adversary chooses the resources to compromise before the protocol starts (static). Assumption (a) is generally true because the links under consideration are not actual physical links that can be monitored. There might be thousands of different ISPs spread over many countries and link monitoring may require observing potentially hundreds (or even thousands) of different possible physical links between every pair of end users, and hence is not considered in the system model.

For a request initiated by a client, say *C*, for resource *R*, a network can be modelled as comprising of three types of entities - the client itself, the intermediate nodes (which forward the lookup message), and the server(s) capable of serving the request. It is assumed that a request for resource *R* can be met by l servers, denoted by $S_{R_1}, S_{R_2}, \ldots, S_{R_l}$. (Here *R* denotes a name or ID that identifies the resource).

[3]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 distributed reputation management mechanisms to ensure that cheating nodes are correctly identified and isolated from receiving services.

A. Proposed Approach

There are two essential building blocks that are exploited in the proposed approach to simultaneously provide both client and server anonymity.

1) A peer-to-peer overlay topology and its associated routing mechanism that utilize intermediate nodes for routing a request (or response) from a client to server (or server to client).
2) An incentive scheme that makes it an optimal strategy for the intermediate nodes to keep information about their neighboring hops secret.

The basic approach is illustrated using an example given in Figure 1, where a request from client C to server S is routed through 10 different intermediate nodes. In this figure, assuming that the above mentioned two building blocks work correctly, none of the nodes from 2 to 9 can know anything about the client or server. Moreover, node 1 upon receiving a request from C, cannot conclude if C is the actual client or just another intermediate node that itself received the request from someone else. (Note that since nodes exist in a virtual overlay topology, where each logical link might translate to several physical links, link sniffing by node 1 to determine whether C originated the request is also very difficult). Furthermore, an indirection layer introduced between nodes 10 and S creates an anonymous communication channel, which shields the identity of S from node 10. The functionality of the indirection layer is achieved in a completely distributed manner without relying on any trusted or centralized entity.

The design of the indirection layer and that of an appropriate incentive scheme are explained in the subsequent sections.

III. Details of the Anonymous Lookup Protocol

The anonymous lookup protocol consist of two phases, as depicted in Figure 2. The first phase is the *server registration phase* and the second phase is the *client lookup phase*. The server registration phase occurs before the client lookup phase. The steps involved in each of these phases is described below. However, before that it is helpful to provide some definitions.

Definition 1: Terminal nodes: Terminal nodes are the Chord successors of the hash values of a resource name. A lookup message from a client is first routed to a terminal node, which then forwards it to the server. It is assumed that there are *t* terminal nodes for each resource. For resource *R*, its terminal nodes are denoted by $T_{R_i}\ \forall i \in \{1, \ldots, t\}$ (for simplicity, it is assumed that *t* is a power of 2). If resource R hashes to Chord ID RI, then the *t* terminal nodes are the Chord successors of the following Chord IDs.

$$(RI + \frac{2^m}{t} * (i-1)) mod(2^m),\ \forall i \in \{1, \ldots, t\} \tag{1}$$

Uniformly locating the terminal nodes around the Chord ring, as given by Equation 1, ensure that the lookup paths to different terminal nodes are as node disjoint as possible.

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 is referred to as a *request chain*. For now, request chains to different terminal nodes of a resource are assumed to be node disjoint, as shown in Figure 2.

Definition 2: Content index node: Content index nodes are the Chord successors of the hash values of the contents of a resource. (In practice, to reduce the cost of computing the hash function, one can use a digest of the contents instead). For

resource *R*, its content index node is denoted by CI_R. The routing of a message from a server S_{R_j}, where $j \in \{1, \ldots, l\}$ to a content index node may go through other intermediate nodes. This list of intermediate nodes along with the server is referred to as a *service chain*. For simplicity, service chains are also assumed to be node disjoint, as shown in Figure 2.

A. Server Registration Phase

Each of the servers, $S_{R_1}, S_{R_2}, \ldots, S_{R_l}$, calculate the hash value of the contents of *R*, i.e., *hash(Content(R))*. For example, if *R* is the name of a file then the input to the hash function is the file itself. The servers then use the Chord lookup protocol to send a registration message $Msg_{register}$ to the content index node CI_R. Intermediate nodes also store the IP address of the node, referred to as the *precedent* node, from which the message was received.

Each registration message $Msg_{register}$ contains the following information - *hash(Content(R))*, resource name or ID (R), marginal cost (MC_{total}).

hash(Content(R)) is used to route the registration message to CI_R, the Chord successor of this value.

The name *R* is used by the intermediate nodes and also CI_R to know about the resource for which registration is being done.

MC_{total} contains S_{R_j}'s marginal cost $MC_{S_{R_j}}$ of providing the resource. An intermediate node on receiving the registration message updates MC_{total} by adding its own marginal cost to the received value.

CI_R receive l such registration messages, and thus knows that resource *R* can be obtained through any of the nodes that sent the registration message. These nodes comprise the set of precedent nodes of CI_R, and are represented by P_{CI_R}.

CI_R then uses the resource name *R* and Equation 1 to locate the corresponding terminal nodes $T_{R_i} \forall i \in \{1, \ldots, t\}$. The terminal nodes are informed by CI_R that resource *R* can be accessed through it.

The nodes in P_{CI_R} do not include the MC_{total} value in the registration message they send to CI_R. Once a request for resource *R* is received by CI_R from the terminal nodes, CI_R holds a second price sealed-bid auction (also called the Vickrey auction) with all its precedent nodes as the bidders. CI_R obtains the resource from the precedent node that offers to provide the resource at the lowest cost. The service chain containing the lowest cost bidder is called the winning service chain *WSC*. It must be noted that MC_{total} represents the minimum price that must be paid by CI_R in order to obtain the resource from the corresponding service chain.

B. Client Lookup Phase

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 all the terminal nodes of resource *R*, such that at most one message is sent out for all the terminal nodes that require going through the same next hop neighbor - the terminal node selected is one which is closest to that neighbor. As a result, the number of terminal nodes that are contacted during a client lookup phase may be less than the total number of terminal nodes for a resource, and therefore, the number of request chains formed, denoted by *k*, are typically less than *t*. Thus, $k \leq t$.

Together the parallel lookup messages towards different terminal nodes constitute a single lookup process initiated by client *C* for resource *R*. Each lookup message Msg_{lookup} contains the following information - address of one of the *k* terminal nodes (T_{R_i}), resource ID (R), maximum price offered (P_C), marginal cost (MC_{total}), request ID $Reqid_{public}$).

$Reqid_{public}$ identifies the lookup process such that CI_R on receiving the resource requests know that they pertain to the same lookup process. Thus, the same value of $Reqid_{public}$ is included in all the lookup messages.

MC_{total} contains *C*'s marginal cost MC_C. An intermediate node upon receiving the lookup message updates MC_{total} by adding its own marginal cost to the received value.

An intermediate node on receiving a lookup message routes it to the next hop neighbor, and this process continues till the message reaches the desired terminal node, which in turn contacts CI_R to obtain the resource. CI_R receive *k* such requests and from the $Reqid_{public}$ values knows that all the requests pertain to the same lookup process. CI_R then holds a second price sealed-bid auction (also called the Vickrey auction) with all the terminal nodes as the bidders. CI_R 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*.

1) Using Vickrey Auction for Resource Trading: As explained above, CI_R holds two separate Vickrey auctions - one with the terminal nodes as the bidders, and the other with the nodes in P_{CI_R} as the bidders.

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 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 [27].

Since, CI_R (the auctioneer) behaves selfishly and tries to maximize its profit, the auction process involving the terminal nodes (precedent set nodes) needs to ensure the following.

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

In view of the above requirements, a two-phase Vickrey auction protocol is used, which is briefly explained below. The detailed analysis of the two-phase Vickrey auction protocol for its robustness and effectiveness can be found in [23].

2) Secure Vickrey Auction to Determine the Winning Terminal Node: The auction process involving the nodes in P_{CI_R} is carried out using exactly the same procedure, except for the fact that the winner now is the one with the lowest bid, i.e., cost value.

The highest and second highest bids are denoted by M_1 and M_2, respectively. The price offered by a terminal node to CI_R is equal to $P_C - MC_{total}$. (On the other hand, the bid offered by a node in P_{CI_R} is simply 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.

CI_R 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 CI_R. 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 the $Reqid_{public}$ value received by the terminal nodes, so that CI_R can determine that the bids pertain to the same lookup process. The received encrypted bids are sent by CI_R 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), CI_R 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 CI_R. The received key values are now sent by CI_R back to all the bidders in message $Msg_{key-reply}$. At the end of this phase, CI_R and all the bidders are able to open the encrypted bids and find out about the highest and second highest bids.

CI_R then sends a message Msg_{cert} to the winning terminal node certifying that it has won the auction. The received certificate is forwarded along the reverse lookup path until 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 - highest bid M_1, second highest bid M_2, total marginal cost MC_{total}. (MC_{total} is received by CI_R in Msg_{bid}). The corresponding Msg_{cert} message in the auction involving the nodes in P_{CI_R} include only the information about the lowest and second lowest bid.

The information in messages Msg_{cert} and Msg_{lookup} ($Msg_{register}$) allow the intermediate nodes, including the winning terminal node, to calculate their reward for being part of the *WRC* (*WSC*). 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 payoffs received by nodes are discussed in the next section.

At the end of the two auctions, the resource is obtained via the lowest cost precedent node (from the server on the *WSC*), and provided to the terminal node on the *WRC*. The terminal node sends the resource to the client along the reverse lookup path. (Since the nodes on *WSC* and *WRC* have to both receive and send the contents of resource *R*, it is assumed that the $Msg_{register}$ and Msg_{lookup} messages also include the size information of resource *R*. This enables the intermediate nodes to calculate their marginal cost values for participating in the lookup transaction. Note that the client can only estimate the size of resource *R*).

The following subsection summarizes the exact sequence of steps followed in the proposed anonymous lookup protocol. For an easy reference, the various messages used during the server registration and client lookup phases, along with the information they contain, are also summarized in Table I and Table II, respectively.

C. Anonymous Lookup Protocol Steps

Server Registration Phase

1) Server(s) with resource *R* register themselves by sending a registration message $Msg_{register}$ to the content index node, which is the Chord successor of *hash(Content(R))*
   - Intermediate nodes update the value of MC_{total} before forwarding the registration message
   - Registration messages reach CI_R, which is the content index node for resource *R*. It must be noted that the registration messages received by CI_R do not include in them the MC_{total} values

   CI_R now knows that resource *R* can be obtained through its precedent nodes, which are represented by P_{CI_R}
2) CI_R uses the resource name *R* to locate the corresponding terminal nodes $T_{R_i} \forall i \in \{1, \ldots, t\}$
3) The terminal nodes are informed that resource *R* can be accessed through CI_R

Client Lookup Phase

4) 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 (involving the terminal nodes)

Phase I

5) Terminal nodes on receiving Msg_{lookup} send Msg_{bid} to CI_R
6) CI_R 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 $Reqid_{public}$
7) Server sends message $Msg_{bid-reply}$ to the terminal nodes
   After the above step the bidders have encrypted copies of all the bids

Phase II

8) Terminal nodes send their secret key to CI_R in message Msg_{key}
9) CI_R replies with a message $Msg_{key-reply}$ distributing the secret keys among the bidders
   At the end of the above Vickrey auction, CI_R knows the maximum price that it can offer for resource *R*. It then solicits bids from its precedent nodes. It must be noted that these bids correspond to the minimum price at which the precedent nodes can provide the resource.

Vickrey Auction (involving the nodes in P_{CI_R})

Phase I

10) Nodes in P_{CI_R} send Msg_{bid} to CI_R
11) CI_R waits for *l* 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 resource name *R*
12) CI_R sends message $Msg_{bid-reply}$ to the bidders
   - After the above step the bidders have encrypted copies of all the bids

Phase II

13) Bidders send their secret key to CI_R in message Msg_{key}
14) CI_R replies with a message $Msg_{key-reply}$ distributing the secret keys among the bidders
15) CI_R sends message Msg_{cert} to the precedent node with the lowest bid value. The receiving node in turn propagates the message along the service chain until the message reaches the server
    By using the contents of messages $Msg_{register}$ and Msg_{cert}, nodes along the *WSC* know the payoff they have to make to their precedent nodes
16) The server, which is part of the *WSC*, supplies the requested resource. The resource is again propagated along the *WSC* until it reaches CI_R
17) CI_R then sends message Msg_{cert} and resource *R* to $T_{R_{WRC}}$. This message, along with the resource, is sent to *C* using the reverse lookup path
18) *C* after keeping its profit share gives the remainder of its initial offer to the next hop node along the *WRC*. The next hop node then keeps its payoff amount and sends the remaining to its next hop, and so on. This process is repeated until CI_R receives a payoff of M_2 from $T_{R_{WRC}}$
19) CI_R gives a payoff of M_2' to the winning precedent node, i.e., the one with the lowest cost of providing the resource. (It is assumed that $M_2 - M_2' > MC_{CI_R}$, thereby giving a net profit to CI_R). Each node along the *WSC* after keeping its payoff amount sends the remaining to its precedent node. This process is repeated till the server that is part of the *WSC* receives its payoff

D. Distributing Reward to Nodes in WRC and WSC

For any node in *WRC*, say *x*, its payoff Pay_x is calculated as follows.

$$Pay_x = MC_x + (\frac{MC_x}{MC_{total}} * (M_1 - M_2)) \tag{2}$$

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.

$$Profit_C = (\frac{MC_C}{MC_{total}} * (M_1 - M_2)) \tag{3}$$

Likewise, let M_1' and M_2' be the lowest and second lowest bid, respectively, in the auction involving the nodes in P_{CI_R}. Then for any node in *WSC*, say *x*, its payoff Pay_x is calculated as follows.

$$Pay_x = MC_x + (\frac{MC_x}{M_1'} * (M_2' - M_1')) \tag{4}$$

The payoff received by CI_R is equal to,

$$Pay_{CI_R} = M_2 - M_2' \tag{5}$$

The example depicted in Figure 3 illustrates the payoffs received by different nodes in *WRC* and *WSC*. Both *WRC* and *WSC* are darkened in the figure. Numbers inside the nodes represent their marginal cost values. The payoffs to the nodes

on *WRC* and *WSC* are also indicated in the figure. For example, payoff to node *1*, which is part of *WRC*, is 13.33 (=10 + (10/30)*(70-60)), and the payoff to node *1'*, which is part of *WSC* is 6.1. *C*'s profit share is 3.33 (= (10/30)*(70-60)). Thus, *C* effectively has to pay only 86.67(=100-10-3.33) for a resource whose utility to it (after deducting the marginal cost) is in fact 90. Therefore, the use of Vickrey auction ensures that everyone, including the client, server, terminal nodes, and intermediate nodes constituting the *WRC* and *WSC* benefit, i.e., earn more than their marginal costs of participating in the lookup process. This potential of earning higher profits motivate nodes to share their resources and forward messages for others.

IV. Anonymity Analysis

In the anonymous lookup protocol the nodes are assumed to be selfish, and in order to maximize their payoffs they have incentive not to reveal information about their precedent nodes (which send the lookup or registration messages) to their next hop neighbors. This is because otherwise the precedent and next hop nodes can directly negotiate among themselves and by-pass the nodes in-between, and consequently there will be less nodes with whom the profit will have to be shared. The incentive-based strategy of lookups in Chord allows us to exploit this inherent property of information hiding, and anonymity is thus naturally provided by the proposed protocol. Also, note that at no point in the operation of the protocol, the identities of the client or server(s) are revealed - none of the messages contain this information. Even the next hop neighbors of *C* (S_{R_i}) do not know that the request was originated (served) at *C* (S_{R_i}). As can be seen the functionality of the indirection layer (as described in Section II-A) is implemented by nodes between the terminal nodes and server nodes. The nodes constituting the indirection layer are configured during the server registration phase.

Unlike in the traditional Chord protocol (or any other DHT based system), where the successor nodes of keys (referred to as the terminal nodes in the proposed protocol) either directly store the key value or the address of a node containing the key value, the terminal nodes are required to store the address of the content index node. This is important as the protocol tries to provide both client and server anonymity, and otherwise server identity is always known to the terminal nodes. One might argue that a single request- and service-chain might also be used (by using a single terminal node and registering the server(s) directly at that node) to provide both anonymity and resource trading. However, in such a scenario both the client and server(s) would have to speculate about the other's offer and also how much the intermediate nodes would charge for routing. Moreover, the intermediate nodes, in order to maximize their profits, would have to speculate about the cost value they should reveal while still ensuring that the offer received by the auctioneer (content index node) from the client side is more than the minimum price asked by the service chain. To avoid such speculations (and counter-speculations) and enable fast resource trading, Vickrey auction is used. Vickrey auction is used on both the client and server side to decide how the resource is eventually priced. In summary, the two layer indexing scheme, and using the content index node, enables the Vickrey auction protocol, and separates the client side of the lookup process from that of the server side.

After the informal reasoning given above, the formal analysis of the anonymity provided by the proposed anonymous lookup protocol is described below. Since the anonymous lookup protocol is symmetric on both client and server side and resembles an "hour-glass" model, only the client anonymity is analyzed. Similar arguments would apply for proving server side anonymity as well. Also, to make the derived equations more tractable, *N* is set to 2^m, where *m* is the number of bits in a Chord ID.

However, the plots given in Figures 6 and 7 are for much smaller values of *N*, and even these small values provide a high degree of anonymity. The commonly used metrics of *Average Anonymity Set* and *Degree of Anonymity*are used to evaluate the strengths of an anonymous system. Notations used in the analysis are summarized in Table 4.

Definition 3: Average Anonymity Set: Anonymity set, represented by *S*, is defined as the set containing all possible initiators of a lookup request as perceived by the adversary set. The average (or expected) anonymity set is the expected value of $|S|$.

Definition 4: Degree of Anonymity: Entropy [26] is used to measure the degree of anonymity of a system. If *X* is a random variable representing the initiator of a lookup chain, then the entropy, $H(X)$ is a measure of the information content of the probability distribution of X. More formally,

$$H(X) \quad = \quad -\sum_{x \in S} Pr(X = x) \log_2 Pr(X = x) \tag{6}$$

Using this definition of entropy, the degree of anonymity on *S* is calculated as,

$$\begin{aligned} deg(S) &= \frac{H(X)_{aposteriori}}{H(X)_{apriori}} \\ &= \frac{-\sum\limits_{x \in S} Pr(X = x) \log_2 Pr(X = x)}{\log_2(N-1)} \end{aligned} \tag{7}$$

The apriori entropy corresponds to the information that the adversary has before observing any lookup request, and therefore it can only exclude itself from the anonymity set. In the following subsections several possible threat models that are relevant for the anonymous lookup protocol are considered.

A. Threat Model A (Single Adversary)

From the perspective of the adversary *A*, the anonymous lookup protocol provides a high degree of anonymity to the client *C*, as summarized by the following theorem.

Theorem 1: The average size of the anonymity set for C is at least $2^m - m(1 - 0.5^m)$

Proof: The average size of the anonymity set is given by,

$$E(|S|) \quad = \quad Pr(L).|S(L)| + Pr(\overline{L}).|S(\overline{L})| \tag{8}$$

Now to derive $Pr(\overline{L})$ the following lemma is used.

Lemma 1: For the lookup initiated by *C*, *A* lies on at most a single request chain.

Proof: The function *d*() takes as input two Chord IDs (or nodes) and returns the Chord distance between them. In other words, $d(xy)$ returns the Chord distance between two nodes *x* and *y*.

Two cases can arise,

Case 1: $d(CA) \geq 2^{m-1}$: Since only one lookup request is sent for all the terminal nodes that are at a Chord distance greater than 2^{m-1} from *C* (all these terminal nodes require going through the $m-1^{th}$ finger table entry), *A* can be on the lookup path to at most one terminal node.

Case 2: $d(CA) < 2^{m-1}$: Let $2^{i-1} \leq d(CA) \leq 2^{j-1}$, i.e., the Chord ID of *A* lies between the i^{th} and j^{th} finger table entry of *C*. *A* can only be on the lookup paths to terminal nodes that lie in the same range, i.e., the i^{th} and j^{th} finger table entry of *C*.

Without loss of generality, let *A* be on the lookup paths to two terminal nodes T_{R_i} and T_{R_j} $(1 \leq i, j \leq k)$, and $d(CT_{R_i}) \leq d(CT_{R_j})$. Then it must be true that $d(CT_{R_i}) \leq d(CA)$ (because the lookup path to a terminal node lying between the i^{th} finger table entry and *A* would not pass through *A*). But since at most a single request is sent out for all the terminal nodes that go through the same next hop neighbor - the terminal node selected is one which is closest to that neighbor. Therefore, *C* sends a request towards only T_{R_i}. Hence, there is a contradiction that *A* is on the lookup path to both T_{R_i} and T_{R_j}. ■

From Lemma 1, it can be concluded that request chains are node disjoint, i.e., at most a single request chain passes through *A*. Therefore, *A* will be on a request chain iff it lies on one of the y_i regions (as indicated in Figure 5). The region y_i is the Chord distance along the clockwise direction between the i^{th} finger of *C* and the terminal node (when there is a terminal node between the i^{th} and $(i+1)^{th}$ fingers), which is closest to that finger. From the property of Chord, the number of hops in the region y_i are $\log(y_i)$. So the probability that *A* lies on one of these regions (i.e., y_is) is given by,

$$Pr(\overline{L}) = \frac{\log(\sum_{i=1}^{i=k} y_i)}{N} = \frac{\log(\sum_{i=1}^{i=k} y_i)}{2^m}$$

For the worst case (best case for the adversary), it can be concluded that

$$\frac{\log(\sum_{i=1}^{i=k} y_i)}{2^m} \leq \frac{\log(\sum_{i=1}^{i=m} y_i)}{2^m} = \frac{\log(N)}{2^m} = \frac{\log(2^m)}{2^m} = \frac{m}{2^m} \tag{9}$$

Now assuming that the adversary lies on one of the request chains and uses its q^{th} finger for routing the request, the following expression for the average size of the anonymity set is derived.

$$\begin{aligned} E(|S|) &= (1 - \frac{m}{2^m})|S(L)| + \frac{m}{2^m}|S(\overline{L})| \\ &= (1 - \frac{m}{2^m})(N-1) + \frac{m}{2^m}2^{m-q} \end{aligned} \tag{10}$$

The average size of the anonymity set is a function of *q*, and the minimum value of $|E(S)|$ is obtained when $q = m$ (adversary uses its m^{th} finger to route the lookup). Substituting in 14, the following lower bound is obtained.

$$E(|S|) \geq 2^m - m(1 - 0.5^m) \tag{11}$$

■

The auctioneer (CI_R) acting as an adversary is a specific case of this threat model, and has similar analysis for the average anonymity set size. The auctioneer only know the identity of the terminal nodes through which it receive the lookup requests.

B. Threat Model B (Multiple Adversaries)

It is possible to have multiple adversaries in a network that can collude, i.e., share their information, in order to determine from where the request originated. Let there be *u* number of such adversaries. The robustness of the proposed protocol against the multiple adversary scenario is demonstrated by the following lemma.

Lemma 2: Adversaries lying on all the request chains cannot collude to further reduce the size of the anonymity set than that available with the most downstream adversary.

Proof: Consider a scenario in which the number of adversaries in the system is so large that there is an adversary on all the *k* request chains. Only the most downstream adversary in each of the request chains is considered, since they are closest to the client. (In a Chord ring, a node has more information about the region of the identifier space that is closer to it than about a far away region). Let $A_1, A_2, \ldots, A_k$ be *k* such adversaries lying on request chains $1, 2, \ldots, k$, respectively. The corresponding anonymity sets and finger table entries used by these adversaries are represented by $S_1, S_2, \ldots, S_k$ and $q_1, q_2, \ldots, q_k$, respectively. Without loss of generality, let $q_1 > q_2 > \ldots > q_k$.

From the property of the Chord routing protocol and also as given in [14], q_1 least significant bits of the adversary A_i are the same as the q_1 least significant bits of the client *C*. Therefore, $|S_1| = 2^{m-q_1}$. Similarly, for any *j*, where $2 <= j <= k$, the q_j least significant bits of the adversary A_j are the same as the q_j least significant bits of the client *C*, and hence $|S_j| = 2^{m-q_j}$. But since $q_1 > q_2 > \ldots > q_k$, these q_j bits would be a suffix of the least significant q_1 bits of A_1.

Now it is easy to see that, $S_1 \bigcap S_2 \bigcap \ldots \bigcap S_k = S_1$. In other words, the most downstream adversary cannot further reduce its anonymity set size by using the information available with the adversaries on other chains. ■

Based on the above observation the expected size of the anonymity set for multiple adversaries is calculated below.

Theorem 2: The average size of the anonymity set is at least $2^m - um(1 - 0.5^m)$

Proof:

$$E(|S|) \quad = \quad Pr(L).|S(L)| + Pr(\overline{L}).|S(\overline{L})| \tag{12}$$

For a large value of N,

$$Pr(L) \quad = \quad (1 - \frac{m}{2^m})^u \tag{13}$$

■

Using the approximation $(1 - x)^u \approx 1 - ux$ the following expression for the average size of the anonymity set is derived.

$$E(|S|) \quad = \quad (1 - \frac{um}{2^m})(N - 1) + \frac{um}{2^m} 2^{m-q} \tag{14}$$

From Lemma 2, the anonymity set of the most downstream adversary (closest) to the client is contained (subset) in the anonymity sets of all the other adversaries. Therefore, q in the above equation is the finger used by the most downstream adversary to route a lookup. Substituting $q = m$, a lower bound on the average size of the anonymity set is obtained.

$$E(|S|) \quad \geq \quad 2^m - um(1 - 0.5^m) \tag{15}$$

C. Threat Model C (Multiple Adversaries Populate the Finger Table of C)

In this threat model, multiple adversaries are the first-hop nodes of *C*. Since from Lemma 1 it can be concluded that the request chains are disjoint, two or more first-hop adversaries receiving a lookup request can accurately identify the client *C*. However, the probability of such an event happening is very low, as shown below.

Lemma 3: The probability of two or more adversaries being the first hop nodes of C is very small.

Proof: From Lemma 1, it can be conluded that the request chains are node disjoint. Therefore, if there are *k* request chains, then there are exactly *k* first-hop possible positions that the adversaries can occupy. Let *X* be the event that two or more adversaries occupy these *k* positions, *Y* be the event that no adversary lies on these *k* first-hop positions, and *Z* be the event that exactly one adversary lies on one of these *k* first-hop positions.

Then,

$$Pr(X) \quad = \quad 1 - Pr(Y) - Pr(Z) \tag{16}$$

For a large value of *N*,

$$Pr(Y) \approx (\frac{N-k}{N})^u \tag{17}$$

$$Pr(Z) \quad = \quad \binom{u}{1}(\frac{k}{N})(\frac{N-k}{N})^{u-1} \tag{18}$$

Therefore,

$$\begin{aligned} Pr(X) \quad &\approx \quad 1 - (\frac{N-k}{N})^u - \binom{u}{1}(\frac{k}{N})(\frac{N-k}{N})^{u-1} \\ &\approx \quad 1 - (1 - \frac{ku}{N}) - \frac{ku}{N}(1 - \frac{(u-1)k}{N}) \\ &\approx \quad \frac{u^2k^2}{N^2} \end{aligned} \tag{19}$$

For large networks, ($N >> ku$) and hence $Pr(X)$ is very small. (The maximum possible value of *k* is only *m*). ■

D. Degree of Anonymity Calculation

Based on the average anonymity set size values calculated in the previous sections, a generalized expression for the degree of anonymity for the proposed anonymous lookup protocol is derived below.

$$deg(S) = \frac{H(X)_{aposteriori}}{H(X)_{apriori}} \approx \frac{\log_2(2^m - um(1-0.5^m))}{\log_2(2^m)} \tag{20}$$

The plots of Equation 20 show that a very high degree of anonymity is achieved even when a significant fraction of the nodes are controlled by adversaries. Figures 6 and 7 show the variation of degree of anonymity with the number of adversaries present in a network of size 1000 and 50000, respectively. For the case when $N = 50000$, $deg(S)$ is as high as 0.8 even when 6% of the nodes are malicious. In information theoretic terms, this means that about 80% of the bits of the client's identity remains hidden from the adversaries. Moreover, the analytical expression for the degree of anonymity is independent of the number of request chains and the number of terminal nodes, with the implication that irrespective of the number of request chains initiated by a client, a very high degree of anonymity is achieved.

V. Protocol Overhead

The proposed protocol incurs some extra over overhead, which is mainly due to the following two reasons.

1) Message communication involved in formation of request chains and service chains.
2) Use of monetary transactions among nodes.
3) Sending of data using multiple hops from the selected server to the client.
4) Computations involved in message encryption and decryption to achieve message non-repudiation

The maximum message processing overhead is incurred by CI_R, which is $O(k + l)$. The message overhead of the client is $O(\log k)$. The number of messages processed by an intermediate node and a server are $O(1)$. The maximum number of nodes involved in the lookup process are $O((k + l) * \log N)$, where *k* and *l* are the number of request chains and service chains, respectively, and $O(\log N)$ is the length of each chain. Thus, it can be seen that the protocol incurs a reasonable overall message overhead.

VI. Related Work

Anonymity for communication systems has been extensively studied, both for client-server and P2P computing models. Most of the existing anonymity protocols are for client-server computing model and hide the identity of the initiator (client). Initiator anonymity is provided using either rerouting based systems or non rerouting based systems. Crowds [16], Mix [4] and Onion-routing [20] employ rerouting based techniques to achieve initiator anonymity. In Crowds, anonymity on the World-Wide-Web (WWW) is provided by grouping the users into large and diverse groups, so that Web-servers are not able to learn the true source of the request. The user or the initiator submits a request which is forwarded to a random member of the crowd, which then forwards the request to the end server. In Mix and Onion-routing, the sender determines the rerouting path and encrypts the route in a layered fashion so that each intermediate node only knows its previous and next hop node.

Schemes which use a single hop intermediate rerouting path include Anonymizer [7] and Lucent Personalized Web Assistant (LPWA) [6]. An example of a non-rerouting based anonymous communication system is DC-Net (Dining Cryptographer's network) [5]. In this case a broadcast medium is used to achieve initiator and responder anonymity, and therefore it suffers from scalability issues.

In P2P publisher-subscriber systems, Freenet [11], Publius [17], FreeHaven [21] are examples of systems which provide publisher anonymity. Freenet is an adaptive P2P network application that permits publication and subscription of data in an anonymous fashion. Publius and FreeHaven use similar strategies for achieving publisher anonymity. While Publius splits the symmetric key used to encrypt and decrypt a document into n shares, FreeHaven splits the document into *n* shares. Any *k* of the *n* peers must be available to reproduce the key (in the first case) and the document (in the second case). While Gnutella [8] anonymizes only queries, GNUnet [13] provides anonymity for both queries and data transfers.

The authors in [15] suggest using a trusted index server for generating the rerouting path. Moreover, there is a substantial amount of overhead involved in encryptions and decryptions. Furthermore, only client anonymity is provided by this protocol. Other existing work [14], [25], [3] that study anonymity in Chord also focus only on client-side anonymity and not server-side anonymity.

The P5 [22] protocol uses a broadcast hierarchy to achieve both sender and receiver anonymity for peer to peer communications. The protocol uses public key cryptography with per-hop encryption and redundant noise packets to achieve a high degree of anonymity, thereby incurring a substantial overhead.

VII. Conclusion and Discussion

In this paper, an anonymous lookup protocol that provides high degree of client and server anonymity (and hence also unlinkability) in P2P based grids is presented. The protocol builds a distributed anonymity infrastructure by implementing an incentive scheme that motivate nodes to participate in the system and also maintain secrecy of the identities of nodes they interact with as part of any transaction. The protocol uses an auction protocol for trading of network resources and ensures that the rewards received by network nodes are maximized if they truthfully follow the protocol steps. Moreover, the protocol is light-weight as it does not rely on any trusted centralized entity or require specialized encryptions to be performed by the nodes.

To the best of the author's knowledge this is the first protocol that uses an incentive strategy to provide sender as well as responder anonymity in large-scale P2P networks. Since the underlying model assumes that all the nodes behave selfishly (and some even maliciously), the authors believe that the protocol is robust enough to be deployable even in large-scale Internet setting.

As future work, the authors would like to explore the applicability of game theoretical concepts to further analyse and improve the proposed anonymous lookup protocol. The authors would also like to explore scenarios when malicious nodes are in fact able to monitor all network links (say when the protocol is deployed to operate in a LAN environment) and the impact on the degree of anonymity that is provided by the protocol.

References

[1] A. Acquisti, R. Dingledine, and P. Syverson. On the Economics of Anonymity, Conference of Financial Cryptography, 2003, pp 84-102.

[2] A. Pfitzman, and M. Waidner. Networks without user observability. Journal of Computer and Security, 1987, vol. 6, no. 2, pp. 158-166.

[3] C. Odonell, and V. Vaikuntanathan. Information Leak in the Chord Lookup Protocol. The Fourth IEEE International Conference on Peer-to-Peer Computing, 2004, pp. 28-35.

[4] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM, 1981, vol. 24, pp. 84-88.

[5] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. Journal of Cryptology, 1988, vol. 1, no. 1, pp. 65-75.

[6] E. Gabber, P. B. Gibbons, D. M. Kristol, Y. Matias, and A. Mayer. Consistent yet Anonymous Web Access with LPWA. Communications of the ACM, 1999, vol. 42, no. 2, pp. 42-47.

[7] E. Gabber, P. B. Gibbons, Y. Matias, and A. Mayer. How to Make Personalized Web Browsing Simple, Secure and Anonymous. Conference of Financial Cryptography, 1997, pp. 17-32.

[8] Gnutella. http://gnutella.wego.com

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. The 2001 ACM SIGCOMM Conference, 2001, pp. 149-160.

[10] I. Foster, and C. Kesselman. The Grid: Blueprint for a New Computing Infrastructure. 2nd Edition, Morgan Kaufmann, 2004.

[11] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. Lecture Notes in Computer Science, 2001, pp. 46-??.

[12] J. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, 2000, pp. 10-29.

[13] K. Bennett, and C. Grothoff. GAP – Practical anonymous networking. Privacy Enhancing Technologies Workshop, 2003, pp. 141-160.

[14] K. J. Kumar, and M. Bansal. Anonymity in Chord. www.cs.berkeley.edu/ kjk/chord-anon.ps, Dec 2002.

[15] L. Xiao, Z. Xu, and X. Zhang. Low-Cost and Reliable Mutual Anonymity Protocols in Peer-to-Peer Networks. IEEE Transactions on Parallel and Distributed Systems, 2003, vol. 14, pp. 829-840.

[16] M. K. Reiter, and A. D. Rubin. Crowds: Anonymity for Web transactions. ACM Transactions on Information and System Security, 1998, pp. 66-92.

[17] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship resistant and source-anonymous web publishing system. USENIX Security Symposium, 2000, pp.59-72.

[18] Napster. http://www.napster.com

[19] N. Nisan. Algorithms for Selfish Agents: Mechanism Design for Distributed Computation. Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, 1999, vol. 1563, Springer, Berlin, pp. 1-17.

[20] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion Routing. IEEE Symposium on Security and Privacy, 1997, pp. 44-54.

[21] R. Dingledine, M. J. Freedman, and D. Molnar. The Free Haven Project: Distributed Anonymous Storage Service. Workshop on Design Issues in Anonymity and Unobservability, 2000, pp. 67-95.

[22] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A Protocol for Scalable Anonymous Communications. IEEE Symposium on Security and Privacy, 2002, pp.58-70.

[23] R. Gupta, and A. K. Somani. A Pricing Strategy for Incentivizing Selfish Nodes To Share Resources In Peer-to-Peer (P2P) Networks. IEEE International Conference on Networks, Singapore, 2004.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. ACM SIGCOMM (San Diego, 2001), 2001, pp.342-356.

[25] S. Hazel, and B. Wiley. Achord: A variant of the chord lookup service for use in censorship resistant peer-to-peer publishing systems. 1st International Workshop on Peer-to-Peer Systems, 2002.

[26] S. Steinbrecher, and S. Kopsell. Modelling Unlinkability. Privacy Enhancing Technologies Workshop, 2003.

[27] T. Sandholm. Limitations of the Vickrey Auction in Computational Multiagent Systems. 2nd International conference on Multi-Agent Systems, 1996, pp. 299-306.

[28] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. Karma: A Secure Economic Framework for Peer-to-Peer Resource Sharing. Workshop on Economics of Peer-to-Peer Systems, 2003.

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

[30] Y. Guan, X. Fu, R. Bettati, and W. Zhao. An Optimal Strategy for Anonymous Communication Protocols. International Conference on Distributed Computing Systems, 2002, pp.257-??

APPENDIX

1) **Chord Overview**

Chord [9] 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 (where *m* is usually 32 or 64) 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. We will use the term "key" to refer to both the original key and its image under the hash function, as its meaning will be clear from the context. Similarly, the term "node" will refer to both the node and its identifier under the hash function.

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 (finger or neighbor) 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.

2) KARMA Protocol

KARMA is a completely distributed protocol for implementing a system of virtual currency in P2P networks. The protocol ensures secure transaction between any pair of nodes. By secure it is meant that a node cannot falsely increase its (and reduce others') currency, and also does not stand to gain from a transaction (process where nodes provide service for some compensation) unless it successfully commits its half of the transaction. KARMA maintains all of its internal state in a peer-to-peer distributed hash table (DHT). The bank-set $Bank_A$ of a node A is a set of k peers that independently maintain the karma balance of that node. KARMA uses the DHT to map nodes to bank-sets. The k closest nodes in the identifier space to the Chord ID of A constitute the bank-set of A. Picking k consecutive hosts for the bank-set allows the secure routing to the bank-set to be performed efficiently.

Each member of $Bank_A$ stores the amount of karma in A's account, signed with A's private key, as well as a transaction log containing recent payments A has made to other nodes. Signing of the balance by A ensures that the value is tamper-resistant. The transaction log acts as proof of A's payment, and comes into play if the other party in the transaction does not send A the file for which the payment was made. The bank-set corresponding to each node also stores - 1) the last used sequence number, which is part of the message sent by a node authorizing its bank-set to transfer karma from its account to the account of some other member (used to eliminate the possibility of replay attacks), and 2) the current epoch number (for periodic currency adjustments and ensuring that per-capita karma in the system is roughly constant). The karma transfer between nodes, say node A to node B, takes place as follows. A first sends to B a signed message authorizing $Bank_A$ to transfer a given amount of karma to B. B forwards this message to its bank-set, who contact $Bank_A$ in turn. If A has sufficient karma in its account to fund the transaction, the amount is deducted from A's account and credited to B's account, and B can proceed with the resource transfer to A. For details about the security aspect of the KARMA protocol one can refer to [28].

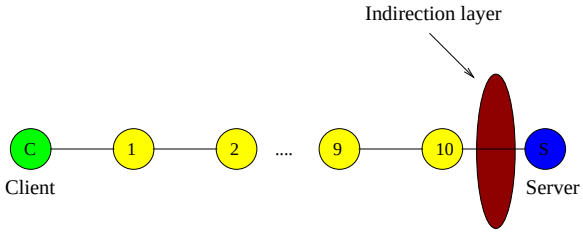

Fig. 1. Figure depicting the intuition behind the anonymity providing strategy.

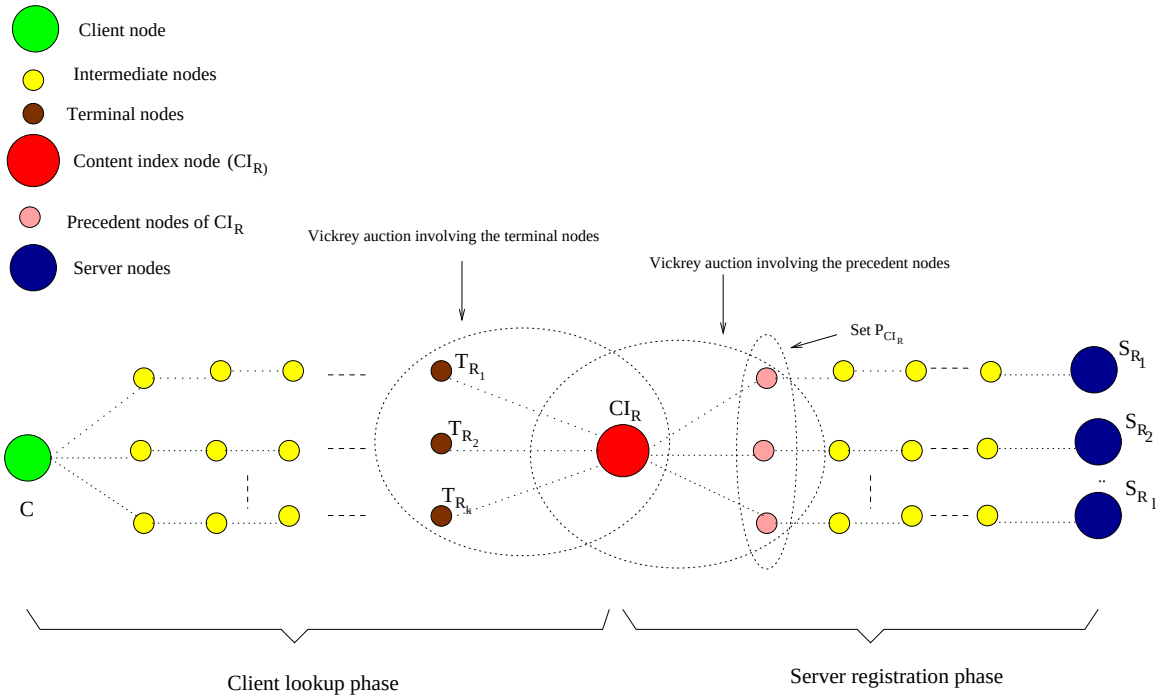

Fig. 2. Figure depicting the operation of the anonymous lookup protocol.

SC_j = j^{th} service chain

RC_i = i^{th} request chain

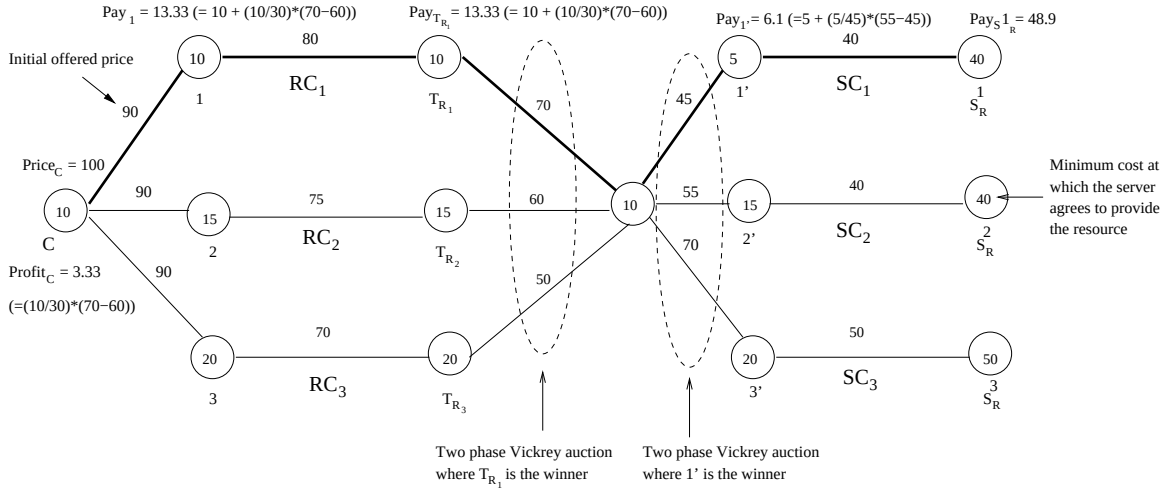

Fig. 3. An example illustrating how the payoffs are distributed among the *WRC* and *WSC* nodes based on their marginal costs.

A:	Adversary
u:	Total number of adversaries (in case of more than one)
$d(ij)$:	Distance (number of identifiers) between nodes i and j
$d_{binary}(ij)$:	Binary representation of distance between nodes i and j
L:	The event that the adversary does not lie on any lookup chain
S(x):	Anonymity set for event x
d(S):	Degree of anonymity on set S

Fig. 4. Notations for anonymity analysis

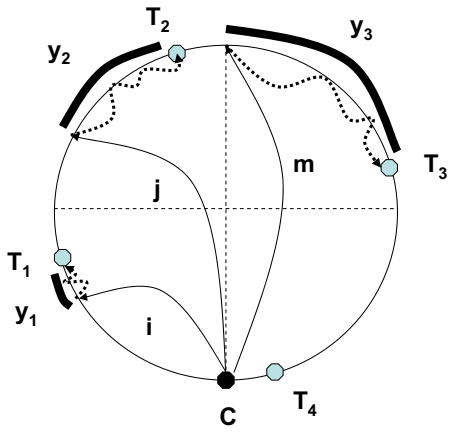

Fig. 5. Multiple request chains initiated by *C*

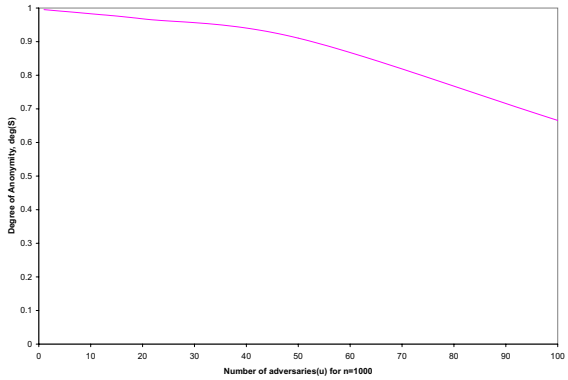

Fig. 6. Variation of degree of anonymity with size of adversary set when $N = 1000$.

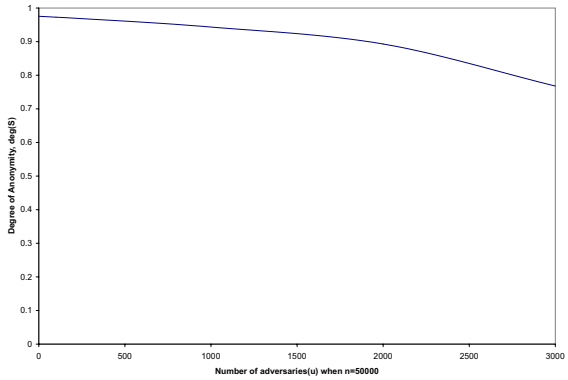

Fig. 7. Variation of degree of anonymity with size of adversary set when $N = 50000$.

$Msg_{register}$	$hash(Content(R)), R, MC_{total}$
Msg_{bid}	$E(randKey_i; b_i), R$
$Msg_{bid-reply}$	$\cup E(randKey_i; b_i), R$
Msg_{key}	$randKey_i, R$
$Msg_{key-reply}$	$\cup randKey_i, R$
Msg_{cert}	M_1', M_2', R

TABLE I

VARIOUS MESSAGES COMPRISING THE SERVER REGISTRATION PHASE AND VICKREY AUCTION INVOLVING THE NODES IN P_{CI_R}. THE RESOURCE NAME R IS INCLUDED IN ALL THE MESSAGES SO THAT THE RECEIVER CAN CORRECTLY ESTABLISH THE CONTEXT FOR THE RECEIVED MESSAGE (FOR EXAMPLE IT IS POSSIBLE FOR CI_R TO BE THE CONTENT INDEX NODE FOR SOME OTHER RESOURCE ALSO). M_1' AND M_2' ARE THE LOWEST AND SECOND LOWEST BIDS, RESPECTIVELY.

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

TABLE II

VARIOUS MESSAGES COMPRISING THE CLIENT LOOKUP PHASE AND VICKREY AUCTION INVOLVING THE TERMINAL NODES. THE VALUE $Reqid_{public}$ IS INCLUDED IN ALL THE MESSAGES SO THAT THE RECEIVER CAN CORRECTLY ESTABLISH THE CONTEXT FOR THE RECEIVED MESSAGE.

PLACE PHOTO HERE

Rohit Gupta is currently working in Amazon.com in Transaction Risk Management group. He earned his PhD degree from Iowa State University in 2005, and his MBA from University of San Francisco in 2001. He received his undergraduate degree in Computer Science and Engineering in 1997 from REC Kurukshetra, India. From 1997 to 1999, he worked as a Research Engineer in Call Processing group, working on ISDN and WLL technologies, in Centre for Development of Telematics (C-DOT), New Delhi India. His research interests include networking, peer-to-peer systems, parallel and distributed computing, and telecommunications.

PLACE PHOTO HERE

Souvik Ray is currently pursuing his PhD in Computer Engineering at Iowa State University. He earned his M.S in Computer Sc from University of Louisiana at Lafayette. He has a B.S in Chemical Engineering from Jadavpur University, Kolkata, India. His research interests include distributed systems security, operating systems and computer architecture.

PLACE PHOTO HERE

Arun K. Somani is currently Jerry R. Junkins Endowed Chair Professor of Electrical and Computer Engineering at Iowa State University. He earned his MSEE and PhD degrees in electrical engineering from the McGill University, Montreal, Canada, in 1983 and 1985, respectively. He worked as Scientific Officer for Govt. of India, New Delhi from 1974 to 1982 and as a faculty member at the University of Washington, Seattle, WA from 1985 to 1997 in electrical engineering and computer science and engineering departments where he was promoted to Full Professor in September 1995.

Professor Somani's research interests are in the area of fault tolerant computing, computer interconnection networks, WDM-based optical networking, and parallel computer system architecture. He is the chief architect of an anti-submarine warfare system (developed for Indian navy) and Meshkin fault-tolerant computer system architecture (developed for the Boeing Company). He has also developed several robust interconnection topologies, architected, designed, and implemented a 46-node multi-computer cluster-based system, Proteus, using a large grain message-passing model and separate data and control planes, and uses fiber optic communication links. His current research is in developing scalable architectures and algorithms to manage, control, and deliver dependable service efficiently for network employing optical fiber technology, wavelength division multiplexing, wavelength conversion, wavelength sharing, traffic grooming, access network design, Fault and Attack Management (FAM) in optical networking.

He has served on several program committees of various conferences in his research areas. He was the General Chair of IEEE Fault Tolerant Computing Symposium - 1997 and Technical Program Committee Chair of International Conference on Computer Communications and Networks, 1999, and OPTICOMM 2003. He is serving as the General Chair of BroadNets 2005. He has served as IEEE distinguished visitor and IEEE distinguished tutorial speaker. He has been elected a Fellow of IEEE for his contributions to theory and applications of computer networks.

PLACE PHOTO HERE

Zhao Zhang is an assistant professor of computer engineering at Iowa State University. His research interests include computer architecture and parallel and distributed systems. He received the BS and MS degrees in computer science from Huazhong University of Science of Technology, China, in 1991 and 1994, respectively, and the PhD degree in computer science from the College of William and Mary in 2002. He is a member of the IEEE and the ACM.