Peer-to-peer auctioning with fairness incentives

Bartłomiej Rozanski

2006
ABSTRACT
In this paper we introduce a protocol for peer-to-peer auc-
tioning with incentives for fair cooperation. The protocol is
based on different assumptions than client-server auction-
ning systems, and utilizes strengths of DHT-based P2P algo-
rithms to alleviate problems of existing protocols. Unlike
present solutions for client-server architecture, our protocol
is scalable to large networks, independent of traffic bottle-
ncks, has smaller operational cost and allows more conve-
nient searching for suitable sale offers.

KEY WORDS
P2P, cooperation, auctions, fairness, incentives

1 Introduction
Online auctioning belongs to the most popular web services
among personal users. In many countries systems such as Ebay form a significant economic market with a huge
number of performed transactions. All of these systems
are client-server based with limited scalability, and therefore the growth of the auctioning system (number of users,
listed items etc.) results in major increase of operational
costs. This in turn leads inevitably to higher fees for listing
items, as well as higher final value fees for end users.

Many of these end users are familiar with distributed
technology and its advantages, such as higher availability
and lower costs. However, it is also well known that P2P
networks lack methods for enforcing fairness which results
in high level of uncertainty of other participants’ honesty.
Therefore, the question whether online auctioning services
can be realized in P2P environment becomes more and
more important.

As we show in this paper, P2P auctioning is not only
possible, but it can also offer features not found in client-
server solutions. We introduce a protocol for peer-to-peer
environment which provides mechanisms for honest bid-
ding and for fair winner resolving. Each transaction be-
tween two nodes is assigned a description which later al-
lows other network participants to judge fairness of both
parties and hence – to tell apart honest and dishonest nodes.
Our solution is not limited to online auctioning but can also
be used with e.g. P2P file sharing to increase user fairness.

The paper is organized as follows: Section 2 describes
requirements for distributed matching, Section 3 introduces
algorithmic tools necessary for our protocol, Section 4
presents the goals and describes the operations of offer-
request matching protocol. Section 5 summarizes the re-
results and outlines open questions and further research prob-
lems.

2 Requirements
Let us start with some observations on online matching and
related differences between the client-server architecture
and large scale P2P networks.

2.1 Observations on online matching

- many matchings are nowadays performed without
  bids (“Buy It Now” option) rather than via time-
  consuming bidding. Therefore, time synchronization
  is less significant, as well as real-time operation. Nor-
mally, these issues cause significant problems with
P2P algorithm design;

- auctioning services cope with huge amount of inform-
ation about users and recent matchings. These cause
significant problems in the case of client-server archi-
tecture. In order to reduce costs some service oper-
ators decide to drop old data records. Although it
helps to save the capacity, it also leads to situations
where user honesty can not be entirely proven. In a
distributed environment with higher data capacity and
more uniformly spread communication load, it is pos-
sible to alleviate the problem;

- bidding period is roughly fixed and in some auction-
ing systems not extended upon arrival of higher bids.
This results in following problem: matchings are per-
formed at the very end of offer period when users try
to place their bids as late as possible ("late rushes").
These often guarantee better outcome for requesting
users – i.e. lower prices paid by auction winners.
However, such strategies also vastly reduce chances
of those users who are not able to place their requests
at this very last moment - e.g. because they can not
connect to auctioning node. Preventing such strate-
gies will improve matching fairness.
2.2 Related work

The subject of online matching in peer-to-peer environment has been studied by Dasgupta and Kalogeraki [2]. Their paper presents a P2P model with buyer and seller agents handling search and bidding operations. The seller agent operates locally and performs bidding depending on buyer agents who arrive at its node. The paper does not consider malicious behavior of users, which in our belief poses significant threat in current P2P systems. Moreover, there are no methods of tracking user fairness, hence to provide incentives for honest cooperation.

Kamvar, Yang & Garcia-Molina [4] proposed a matching system for unstructured P2P networks with built-in incentives for cooperation. Similar to [2], their solution offers no means of protection against malicious users. Above that, it is strictly limited to unstructured P2P networks (like Napster/Gnutella), and cannot be implemented in more efficient DHT-based peer-to-peer networks.

3 Tools

3.1 URE-Onions

Onion routing technique (see [6]) is relatively simple yet powerful solution for anonymous communication over distributed networks. Here is the basic mechanism.

To send message \( m \) to \( D \), node \( S \) chooses randomly some anonymity path of intermediate nodes \( J_1, \ldots, J_\lambda \). It constructs an onion encoding message \( m \) as:

\[
E_{J_1}(E_{J_2}(...(E_{J_\lambda}(m, D), J_\lambda), J_\lambda), J_2)
\]

where \( E_X(m) \) denotes the encryption of \( m \) with public key of \( X \) using some probabilistic scheme.

This onion is first sent by \( S \) to \( J_1 \). Upon receipt node \( J_1 \) decrypts the ciphertext and obtains plaintext consisting of two parts: an onion with one layer “peeled off” and the identity of subsequent path node – \( J_2 \). It forwards the new onion to \( J_2 \) who in turn decrypts and sends it to \( J_3 \) and so on. After \( \lambda \) steps the onion reaches its destination \( D \) and the recipient can read the message.

Universal Re-Encryption is an extension of the ElGamal scheme. An URE-ciphertext of message \( m \) is computed by the node \( P \) as a quadruple:

\[
(\alpha_0, \beta_0; \alpha_1, \beta_1) = (m \cdot y^{k_0}, g^{k_0}, y^{k_1}, g^{k_1})
\]

with \( k_0 \) and \( k_1 \) picked randomly. In fact, this is a pair of ElGamal ciphertexts of messages: \( m \) and 1. The advantage over plain ElGamal scheme is that everybody can re-encrypt such ciphertext without any additional knowledge. Moreover, with URE scheme the identity of \( P \) remains hidden, but it is possible to re-encrypt its ciphertexts. Detailed description of URE scheme can be found in [3].

An URE-Onion is built with \( \lambda \) separate URE-ciphertexts:

\[
E_{x_1} + \cdots + E_{x_{\lambda}}(J_{i+1}), 1 \leq i \leq \lambda - 1
\]

\[
E_{x_1} + \cdots + E_{x_{\lambda}}(m), i = \lambda
\]

The messages for different routing steps are contained in separate blocks (ciphertexts). Upon receipt of an URE-Onion each node partially decrypts and then re-encrypts all blocks. After decryption \( J_i \) can read subsequent destination \( J_{i+1} \) in one of the blocks. Next, after re-encryption step \( J_i \) permutes all blocks randomly and forwards the onion to \( J_{i+1} \). Observe that any two onions containing given message \( m \) will look thoroughly different which provides protection against repetitive attacks.

A special case are the URE-Onions encoding message 1 – so called navigators:

\[
(\alpha_0, \beta_0; \alpha_1, \beta_1) := (y^{k_0}, g^{k_0}, y^{k_1}, g^{k_1})
\]

Navigators serve as a kind of envelope – knowing navigator of node \( P \), \( S \) can insert a message into it so that only \( P \) and no one else will eventually read it after the navigator traverses its path. Insertion of the message is done by simple multiplication of the first component by the \( m \) and does not require knowledge of neither public nor private key of \( P \). Moreover, it does not reveal identity of sender \( S \).

3.2 Hiding Data Sources

In [5] a protocol for anonymous data retrieval was presented. The idea behind the protocol is to use access points to serve data instead of allowing direct access to nodes holding respective files. As a result actual sources remain hidden but the data can be accessed by the protocol participants.

![Figure 1. Access structure](image)

To provide access to file \( x \) its holder \( P \) constructs anonymity paths from a number of access points \( A_i(x) \) to itself. The addresses of the points are determined from a hash function \( H(x, i) \) which is known and computable to all network participants. Each \( A_i \) receives a navigator from \( P \) which will later allow it to pass user’s request for file \( x \).

To obtain file \( x \) user \( U \) needs to contact any of \( A_i \). The access point embeds the request into navigator from \( P \)
3.3 String-DHT

String-DHT [1] is a mechanism for distributed data storage and retrieval. It allows users in large scale P2P networks to specify data by single or multiple string attributes. The matching algorithm supports searches based on exact text values as well as on prefixes and suffixes.

An attribute is a pair \((a_i \Rightarrow v(a_i))\) where \(a_i\) is a string name of the attribute and \(v(a_i)\) is the value. Exemplary attributes for auctioning are: \("\text{category} \Rightarrow \text{photo}\)\), \("\text{producer} \Rightarrow \text{Canon}\)\), \("\text{model} \Rightarrow \text{EOS 10D}\)\).

Attributes allow users to precisely describe data they publish in the network. An object (file, database record, etc.) with aforementioned attributes is in String-DHT submitted to the nodes responsible for keys "category", "producer" and "model". Each of these nodes adds the publisher id and the respective attribute value to its list of submissions.

To fetch data matching desired criteria (list of attributes, e.g. \("\text{category} \Rightarrow \text{photo}\), \("\text{model} \Rightarrow \text{EOS 10D}\)\)) user contacts nodes responsible for particular attributes, who in turn reply with the list of submissions matching attribute value \((model = "EOS 10D", etc.)\). If the search is based on multiple attributes the user takes the intersection of received lists as the result. Each resulting submission contains the id of its publisher, hence the user is able to directly access the data.

In addition String-DHT also supports prefix and suffix based searches. This fact, combined with the ability to use multiple criteria allows users to define search queries with various accuracy – they can either define the scope very precisely, or use coarse criteria to “browse” data.

4 Solution

Our protocol for offer-request matching requires some implemented underlying protocol which translates node and file identifiers into network addresses and allows fast lookups. Since there is plenty of known solutions like Chord, CAN or Pastry – we do not describe them here in details. Our only requirement for the lookup protocol is to support searches based on string attributes via String-DHT.

We also assume that every network participant possesses a pair of unique public and private keys and that there is some proxy signature scheme. The later enables designated user \(P\) to compute proxy signature of matching system \(A\) of some given message.

Finally, we take some distributed or client-server based “matching system” as given. This system is responsible for assigning guard nodes to the offers in anonymous fashion and for setting proxy signature scheme with these nodes.

4.1 Definitions

By matching system we will understand a given solution which performs certain operations as described further in this section.

An offer is a submission of some ware or virtual “something” which the user called seller is ready to give or serve. Seller defines a period of time when the offer remains valid. A buyer is the user who requests the good offered by seller within defined period of validity and who places his bid – some measurable return value he is ready to “pay” for requested good. After the offer period expires, the matching system selects the winner – the buyer that offered the highest bid. A transaction can be defined as the relation between offers and buyers – it matches the seller and the winner.

Each transaction can be assigned an opinion – some additional information (rating or memo record) which describes mutual judgments of the transaction sides. Opinions are attached to the transaction with so called tokens that form records of the distributed user behavior database for the matching protocol.

4.2 Goals

Let us now state the goals of the protocol from the point of view of different users.

all participants – eventual transaction will be valid and guaranteed by the matching system \(A\). The transaction token will store mutual opinions of both transaction parties – if such were added;

buyers – matching is won by a node who placed the maximum bid, not by someone chosen arbitrarily by the seller or by any other node;

seller – matching system picks the maximum bidder as a winner.

4.3 Operations

Let us now describe the operations supported by protocol participants.

User registration Each new user needs to register to the system \(A\). The registration is necessary to assign to user \(W\) his guard – \(G(W)\). The guard will be responsible for storing tokens of \(W\). Thus, \(G(W)\) will be the node to be contacted (via anonymity paths) in order to fetch information about \(W\)’s history, and evaluate his fairness.
Adding an offer  Let us assume node $S$ wants to offer some good $V$. The goal of this operation is to provide potential buyers access to offer $V$ in such way that they will have no doubts about offer validity. Adding an offer involves following steps:

1. $S$ informs matching system $A$ about the offer $V$;
2. matching system $A$ picks randomly some $P(V)$ to take care of $V$. To enable valid $P(V)$ operation they set up proxy signature parameters which will later allow $P(V)$ to remain anonymous while representing $A$;
3. $P(V)$ determines a reasonably large list of nodes which may possibly be asked for offers similar to $V$ – i.e. if the subject of offer is "Canon EOS 10D Digital Camera", it is an "used item" and the seller is located in “Banff, Alberta / Canada", it may be reasonable to inform nodes responsible for keys "Canon", "camera", "10D" and possibly “Banff”. For practical purposes, some set of standardized rules of offer description should be used in order to simplify user searches. These rule sets can be acquired e.g. via software updates of P2P protocol client, and optionally may include multilanguage schemes.

For each node $A_j(V)$ from the list $P(V)$ builds anonymity path that allows $A_j$ to communicate and forward future buyer’s request for the offer;

4. $P(V)$ sends offer description to all $A_j$.

Searching for offers  Let $W$ be a user looking for some particular content. $W$ selects some keywords describing precisely the subject of interest, and uses the lookup protocol to find offers matching the description.

Note that this solution shifts the entire effort of collecting offers towards individual users. This is different than client-server protocols, where there is a single place to store “fresh” copy of offer list which is operated by single party (central server). Apart from being more suitable for P2P networks, our algorithm has a significant advantage over client-server solution – it allows users to freely define the scope of the search and to fetch results they want with arbitrarily high accuracy.

Gathering knowledge about other users  User $B$ who finds an offer matching his request may want to check trustworthiness of respective seller $S$. To do so he sends a request to node $G(S)$ for some tokens of past $S$’s transactions. After receiving these tokens $B$ can check other users’ judgments of $S$.

Coarse offer value  User $B$ may also want to roughly check the status of the offer – e.g. in order to decide whether it is still reasonable to place his own bid or whether his already placed bid have chances to be accepted. $B$ asks $G(V)$ about the offer value and compares it with his own bid.

To discourage users from placing their bids at the very last moment, the return value can not be given explicitly – i.e. $G(V)$ must not reveal the highest bid. Instead, $G(V)$ sends second highest bid (Viceroy auctioning) or even some older one. While this is quite inaccurate estimation for user $B$, under some circumstances it is sufficient to prove him that the offer value exceeded his bid. Or it provides $B$ with rough information whether his bid has any chance to be accepted by the system – if it is higher than $G(V)$’s reply.

Placing bids  When user $B_i$ decides to take part in bidding of the $V$ offer of seller $S$ he performs following steps:

1. $B_i$ decides on the value of his bid $L_i$ and picks some random $c_i$;
2. $B_i$ builds an anonymity path to some $A_j(V)$;
3. using the anonymity path he sends following tuple to $A_j(V)$:
   \[(L_i, H(c_i), E_S(B_i), E_{B_i}(c_i))\]
   consisting of his bid, commitment to $c_i$ and his identity.
4. at the same time $B_i$ also sends $G(V)$ following pair:
   \[(H(c_i), E_S(L_i||B_i))\]
   which will allow him to prove guilt of $P(V)$ (with the help of seller) if $P(V)$ behaves unfairly;
5. if $L_i$ does not exceed the highest bid known to $A_j(V)$, the access point drops $B_i$’s request. Otherwise it sends following tuple:
   \[b_i = (L_i, t_i, H(c_i), E_S(B_i), E_{B_i}(c_i))\]
   to $P(V)$ where $t_i$ is the time of bid arrival;
6. $P(V)$ replies with a proxy signature $PSig_A(b_i)$ to $B_i$ to confirm the acceptance of the bid;
7. if $L_i$ exceeds the highest bid, the node who placed the latter is informed by $P(V)$ about the change of situation (cancellation). Sale value $\hat{L}$ is then updated and $P(V)$ sends following tuple: $v = (V, L_i, \hat{L}, PSig_A(EB_i(c_i)), a_i)$ to $G(V)$, where $a_i = PSig_A(V||H(c_i)||S||\hat{L})$ is a proxy signature under a message consisting of all data related to the transaction concatenated.

$v$ tuple forms a kind of winning record where the highest bid, sale price, winner identity (covert but guaranteed by matching system) are stored. In addition, $a_i$ value binds: the seller, the buyer and the sale price. $v$ will be later used to construct transaction token which will additionally store mutual judgments of both parties.
Figure 2 depicts messages sent during bid placement.

![Diagram of bid placement]

Note that in step 7 node $P(V)$ sends both $L_i$ and $\bar{L}$. This is because in some auctioning models the highest bid and sale value are not the same – for example in Vickeroy model the winner pays the price of the second bidder.

Observe also that from $B_i$’s point of view the bid is not valid until he has not received acceptance confirmation from $P(V)$. Should $P(V)$ behave unfair and discard $B_i$’s bid, there is additional “assurance” message sent to $G(V)$ (in step 4) that allows $B_i$ to eventually prove $P(V)$ fraud. $B_i$ may be unable to do so if $G(V)$ and $P(V)$ cooperate – but since both nodes are chosen randomly by matching system $A$, the probability of them being anyhow correlated is marginal. $S$ will obviously cooperate with $B_i$ in this step, because it is their common interest to ensure that node with the highest bid wins (see 4.2 Goals).

In addition, let us observe that user’s bid is dropped if it is smaller than maximum bid known to $A_j(V)$ at the moment. Such bids would have no chance of winning anyway, hence dropping does not threaten fairness. On the other hand it reduces the number of messages sent via anonymity paths, which in turn allows protection against DoS attacks towards $P(V)$.

Unfortunately, the procedure also affects fair user, who does not know whether the lack of acceptance message is caused by malicious behavior of some anonymity path node, or simply by the fact that the bid was dropped. In such case $B_i$ should fetch coarse offer value, and – depending on the reply from $G(V)$ – try bidding through another $A_k(V)$ or give up.

Finally, note that until $B_i$ reveals $c_i$ neither guard $G(V)$, $A_j(V)$, nor $P(V)$ know to whose bid apply the interchanged messages. The only node who can check $B_i$’s identity is the seller $S$ who may potentially be interested in buyers reputation. Despite the fact that $B_i$ remains anonymous through the entire bid placement operation, his identity is fixed in $v$ tuple, confirmed (in step 6) by matching system and secure against forgery by another buyer. The later follows from the fact, that $v$ includes unforgeable proxy-signatures of $A$ of two messages: $V||H(c_i)||S||\bar{L}$ $E_{B_i}(c_i)$. First message contains buyer’s $B_i$ commitment to $c_i$ value, while the second provides him with a proof of identity (since no other node can compute $B_i$’s signature for $c_i$). Eventually, these two values allow $B_i$ to prove his win to $G(V)$.

Resolving the winner When the offer period expires all bidders who had not received cancellation message from $P(V)$ contact $G(V)$ to prove their win. To determine which of them actually won the bidding following procedure is executed:

1. $G(V)$ publishes commitment (e.g. hash value) to $v$;
2. bidders publish commitments to their $PSig_A(b_i)$;
3. $G(V)$ reveals $v$;
4. the winner(s) reveals $PSig_A(b_i)$ and everybody can verify the $E_{B_i}(c_i)$ to see that $B_i$ is indeed the node indicated in $v$.

In case of “Buy it Now” option user simply contacts $G(V)$ upon receipt of $P(V)$ confirmation and proves his identity.

Token generation Knowing the identity of the winner, $G(V)$ uses $v$ to construct the transaction token:

$$(V, B_i, S, \bar{L}, PSig_A(V||H(c_i)||S||\bar{L}), PSig_A(E_{B_i}(c_i)))$$

Proxy-signatures of $A$ provide the warranty of token validity. As discussed earlier in this section, both these signatures prove the identities of the offer winner, the seller and of the transaction value.

At the time the token is “in-blanco” and contains only the proof that the matching between $B_i$ and $S$ has been confirmed by $A$. Nevertheless, it is sent to $G(S)$ and $G(B_i)$. Some time after the matching is done both parties will want to attach to the token their own comments or judgments about the transaction. To do so, they will send respective information to the guards who have the token. Since the guards can see who is eligible to supplement the token, they accept or deny the information. At any moment, completed or not, the token can be handed to other nodes requesting information about $B_i$ or $S$.

4.4 Features

To conclude this section let us briefly summarize the features of protocol operations in terms of aforementioned goals.
Transaction validity Transaction validity means that matching of the seller and the auction winner is not forged by any party, but is approved by the matching system. In our scheme this issue is guaranteed by the proxy-signatures of A, which commit the buyer, the seller and the offer value.

Maximum bid rule From the seller point of view steps 3 and 4 of bid placement operation ensure proper bidding outcome of the offer. It is so, since fair $P(V)$ guarantees that the maximum bid within offer period will be chosen. Note that user requests sent to $P(V)$ can be properly ordered by the time of their arrival and by their amount (because it all happens at this single node).

On the other hand, $B_i$ always sends the information about his bid value (and a commitment to it) to $G(V)$, whom $S$ can ask to check what bids have been committed by the buyers. Therefore, seller $S$ is also able to track the offer value.

4.5 Modifications

Multiple winners It is possible to enhance the scheme and allow multiple offer winners. For this purpose protocol operations must allow multiple $v$’s (as defined in step 7 of bid placement operation) instead of only one. Since each such $v$ is built with information that is accepted (proxy-signed) by $A$, no concerns about validity of offer values occur, because it is the matching system that decides how and which $v$’s the $G(V)$ should update.

Seller anonymity Observe that the seller is publicly known from the very beginning of the offer period. Consequently, all nodes performing operations connected to $V$ offer know whose offer it is. In particular, malicious nodes can alter the execution of their operations to do harm to $S$.

Under circumstances it may be beneficial to introduce seller anonymity into the protocol. In a basic form this is a simple task. Namely it suffices for $S$ to:

1. pick some $d_S$ and send $H(d_S)$ to $A$ in step 1 of adding an offer;
2. reveal $d_S$ in step 5 of resolving the winner procedure;
3. remove $E_S(B_i)$ from all operations.

Observe that 3 collides with the maximum bid rule. This results from the fact that now $B_i$ is not able to prove $P(V)$ fraud and hence there is no warranty that $P(V)$ can not choose some bidder that offered a smaller offer value.

To alleviate this problem $S$ should generate a pair of keys for asymmetric cryptography and construct some anonymity path to itself. Then it sends $A$ the address of access point of the anonymity path and the public key. Then, being the only node with respective private key, $S$ will be able to help $B_i$ to prove frauds. $B_i$ will be able to communicate with $S$ via indicated anonymity path.

The unsolved problem in this case is the lack of method to verify the seller’s honesty from the buyer’s point of view.

5 Conclusion

We have shown a P2P based protocol for matching user offers and requests that can be used e.g. for file sharing or internet auctioning. Proposed protocol is based on different assumptions than currently operating client-server solutions, which in our belief makes it more suitable for distributed world wide architectures. Unlike client-server systems, our P2P based protocol is easily scalable and resistant against local attacks or failures. In addition, it requires no costs from users, because no dedicated hardware, connections nor means of security are necessary. Finally, the P2P solution supports advanced search queries which allow users to find offers using flexible criteria and with arbitrarily high precision.

An interesting question, worth of further research, is whether the same network of participants can execute “matching system” protocol and provide all its functionalities without harming either security or anonymity and honesty.

Acknowledgement

Partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).

References