Decentralized Auction-based Pricing with PeerMart

David Hausheer\textsuperscript{1}, Burkhard Stiller\textsuperscript{2,1}

\textsuperscript{1}Computer Engineering and Networks Laboratory TIK  
Swiss Federal Institute of Technology ETH Zurich
\textsuperscript{2}Department of Informatics IFI, University of Zurich  
E-Mail: [hausheer, stiller]@tik.ee.ethz.ch

Motivation and Goals  
PeerMart Requirements and Design  
Analytical and Experimental Results  
Conclusions
Motivation

- **Centralized auctions**
  - Dedicated (centralized) broker
  - High reliability
  - High technical efficiency
  - Not scalable
  - Single Point of Failure
  - Vulnerable against attacks

- **Decentralized (P2P) auctions**
  - Peers act as brokers, providers, and consumers at the same time
  - Scalable
  - No Single Point of Failure
  - Vulnerable against selfish or malicious peers
  - Technical feasibility?
Goals

Key goals

- To maintain the economic efficiency of auctions
- To exploit the scalability and resilience of P2P networks
- To achieve a high reliability even in the presence of malicious peers
- To provide a technically efficient solution

- Re-use of economic efficiency characteristics of double auctions
- Focus on technical feasibility of implementing a P2P double auction in a P2P environment
PeerMart Requirements

☐ Economic efficiency
  • Double auctions close to the ideal case (maximization of benefits)

☐ Technical performance
  • Efficient use of resources (capacity, storage, processing, messages)

☐ Scalability
  • Any number of services under any type of load being traded

☐ Reliability
  • Continuous availability, correctness, secure, DoS-safe, selfish-safe

☐ Accountability

☐ Privacy

☐ Incentive compatibility

☐ Determine suitable trade-offs for contradicting requirements
Design Space for Auctions

Market Institutions

- Quasi-Auctions
  - Walrasian
  - Tâtonnement
  - Shapley-Shubik
    - 1977

- Unified
  - Discriminatory/Fragmented
    - McAfee (1989)

- 2-Sided Auctions
  - Discrete-time
    - Clearinghouse (CH)
  - Continuous-time
    - Double Auction (DA)

1-Sided Auctions

- English
- Dutch

- 1-step (PQ)
- N-step ED
- Q only (CHQ)
- K-DA
- BBDA

- Variants: Walrasian, Linear ED, N-step ED, 1-step (PQ)
- Hybrids: DD, UPDA, Synchronized
- Variants: Computerized or Oral, Privileged Traders, NYSE, CBOT, Globex

Asymmetric, not optimal

Simple but not scalable

Related Work on P2P Auctions

  • Offers/Bids are broadcasted in a Gnutella-like fashion
  • Any peer can answer with a counter offer
  • Not scalable, not strategy-proof

  • Agents are connected in a random network
  • Single agent is assigned as cluster center
  • Cluster size is limited => not scalable
  • No message delay is assumed => not realistic

  • Based on Law Governed Interaction (LGI) paradigm
  • Only the auction process itself is decentralized
  • Communication overhead due to routing messages via controllers
PeerMart Approach: Distributed Double Auctions

- **Key Idea**
  - Providers and consumers offer prices for services
  - Offers are optimally matched by broker peers

- **Basic Algorithm**
  - Providers publish services they wish to provide
    - Broker peers reply with a bid price (current highest pay price)
  - Consumers request services they wish to use
    - Broker peers reply by an ask price (current lowest sell price)
  - Brokers run a matching strategy at regular intervals:
    - Upon every price offer, if offer is higher (lower) than current bid price (ask price) => no match, store offer in a table
    - Otherwise, forward offer to the peer with the best counteroffer (highest bid/lowest ask)
PeerMart uses a structured P2P overlay network to distribute the broker load

- Pastry is used as underlying infrastructure
  - Provides unique 128-bit nodeIds
  - Stores $n$ closest nodeIds (leaf-set) in routing table

Each peer has a public/private key pair to sign messages

- Assumption: Public keys certified offline and bound to nodeIds

Services are mapped onto a redundant set of $n$ broker peers (broker-set)

- Assumption: Services have unique IDs (e.g., hash value of a file)
Technical Design (2)

- Consumers/providers randomly select $f$ brokers out of the broker set
  => Load is uniformly distributed on all broker peers

- Each broker peer keeps table of $m$ highest bids/lowest asks
  - Lower bids and higher asks are rejected

- Matching is performed in a decentralized way
  - Candidates for a match are forwarded to other brokers
    - Candidates: local matches and next best offer
  - Final matches are determined using majority decisions
  - Winners are notified by the $f$ brokers that received the offer
Double Auction in PeerMart

Consumer peer C1 requests service X

Broker peers responsible for service X

Provider peers P1 & P2 offer service X

Exchange candidates

Notify peers

Search BrokerSet using Pastry

ServiceID X

Bid 3 €

Ask 2 €

Ask 1 €

Return BrokerSet

Table of bids/offers

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Bid 3 €

Ask 2 €

Ask 1 €

Exchange candidates
Analytical Results (1)

- Basic effort for maintaining Pastry is $O(\log_b N)$
  - $N$: Number of peers in the network

- Finding responsible broker-set is $O(\log_b N)$

- All subsequent communication happens directly
  - Sending offers is $O(f)$
  - Exchange/forward of matching candidates is $O(f*n)$
  - Notify peers about final matches is $O(f)$
  - $f$: Number of parallel active brokers within a single set
  - $n$: Average broker set size

- Average number of offers stored per peer: $s \cdot n \cdot m$
  - $s$: Average number of service involvements per peer (naturally limited)
  - $m$: Maximum number of offers and bids per service
Analytical Results (2)

Reliability

- PeerMart correctly matches offer pairs for < 50% malicious peers using, e.g., 8 brokers in parallel (out of 64 brokers in total)

- If fraction of malicious peers is < 25%, even 4 parallel brokers provide good reliability
Experimental Setup

- Prototype implemented in Java on top of FreePastry
- Assumptions for experiments:
  - Each peer assigned to \( s = 2 \) services
  - Peers follow the ZIP bidding strategy:
    - Consumer’s price offer:
      \[
      p_{bid} = \min(p_{ask} + \alpha(p_{max} - p_{ask}), p_{max})
      \]
    - Provider’s price offer:
      \[
      p_{ask} = \max(p_{bid} - \alpha(p_{bid} - p_{min}), p_{min})
      \]
    - \( p_{max}, p_{min} \) are reservation prices
      → normally distributed such that
      50% of offers match, \( \alpha \) set to 0.1
  - Malicious peers:
    - Uniformly distributed
    - Do not forward offers nor notify peers

LAN with random latency (100-200 ms)
Experimental Results (1)

Scalability (Increasing Network Size, Message Overhead)

Measured security overhead:
- Overhead for SHA1 with DSA signature is ~15% of message size
- Signing / verifying messages takes 10 - 20 ms (~10% of the average RTT)
on Pentium 4 CPU 2.4 GHz, 512 MB RAM, Java VM 1.4.2

* Message size is ~1 kB

© 2005 D. Hausheer, B. Stiller, ETHZ-TIK
Experimental Results (2)

Efficiency (Redundancy Costs, Message Overhead)

Measured security overhead:
- Overhead for SHA1 with DSA signature is ~15% of message size
- Signing / verifying messages takes 10 - 20 ms (~10% of the average RTT) on Pentium 4 CPU 2.4 GHz, 512 MB RAM, Java VM 1.4.2

* Message size is ~1 kB
Experimental Results (3)

Reliability

- Matches analytical results quite well

![Graph showing Correctly matched offer pairs (%) vs. Malicious broker peers (%) with n = 64]

Increasing redundancy
Summary, Conclusions, and Future Work

- PeerMart defines a fully decentralized double auction
  - Applicable by any peer to trade any service
- Technically designed on top of a P2P overlay network
  - Implemented prototype uses FreePastry as underlying infrastructure
- PeerMart is both technically and economically efficient
  - Scales well even for a large number of peers trading services
- High reliability even in the presence of malicious peers
  - Achieved through redundancy

Future Work
- Punish peers that do not stay to an offered price (=> use Reputation)
- Study other forms of malicious attacks (e.g., DDoS)
- Run prototype in a real-world environment (e.g., PlanetLab)
Thank you for your attention!

=> Further information: http://www.peermart.net/