

TU Darmstadt, March 4, 2005

PeerMart

Secure Decentralized Pricing and Accounting in Peer-to-Peer Networks

David Hausheer

*Computer Engineering and Networks Laboratory, TIK
Swiss Federal Institute of Technology, ETH Zürich
E-Mail: hausheer@tik.ee.ethz.ch*



Outline

- Motivation & Problem Statement
- Goals and Requirements
- Auction-based Pricing with PeerMart
 - P2P Market Model & Main Problems
 - Design Space for Pricing in P2P
 - PeerMart's Approach
- PeerMint: Accounting for Peer-to-Peer Applications
 - Remote Accounting Concept
 - PeerMint's Approach
- Preliminary Conclusions and Future Work

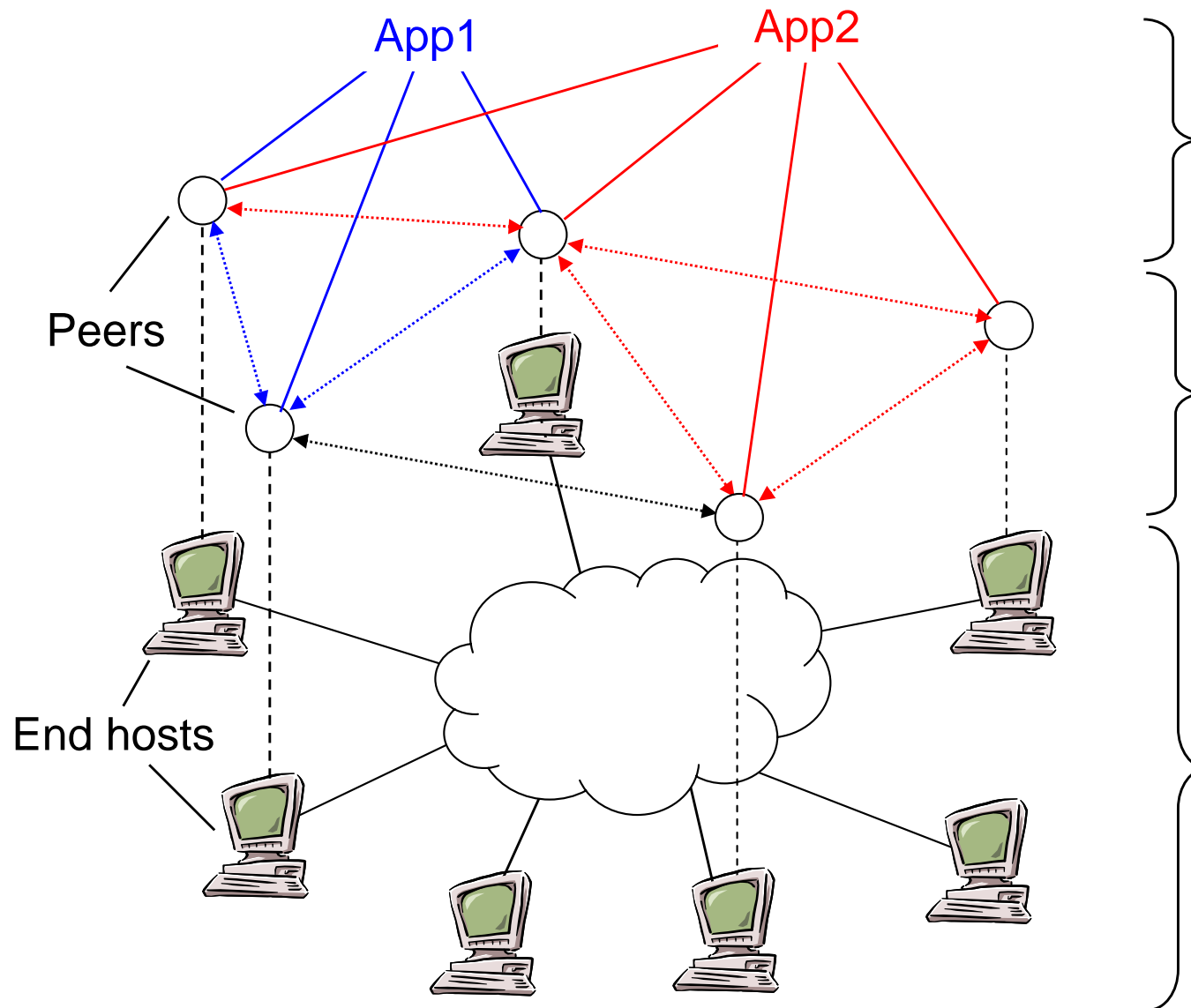


Motivation

- More and more applications benefit from P2P systems
 - **File sharing** systems (BitTorrent, eMule, KaZaA)
 - Distributed **storage**, distributed **computing** (OceanStore, SETI@home)
 - **Host sharing** systems (PlanetLab)
 - PeerCast, Skype, Groove, ...
- General advantages of P2P systems:
 - High **performance**
 - Resources can be aggregated (e.g. parallel downloads)
 - Possibility of load balancing over several nodes
 - **Robustness**
 - No Single Point of Failure
 - Fault tolerance against unreliable nodes
 - **Extensibility, scalability**
 - Resources can be added easily
 - Scalable routing mechanisms exist (e.g. Pastry, Chord)



P2P Architecture



P2P Applications
e.g. Kademia, eDonkey
PeerMart

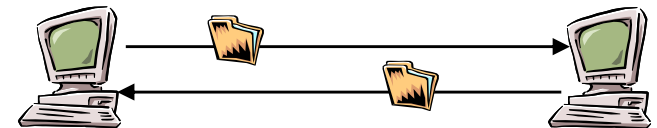
P2P Overlay Networks
e.g. random, structured,
hybrid, centralized

Internet



Current Weaknesses

- It is assumed that peers **cooperate** (share their resources)
 - However, peers are **autonomous** => cannot be forced to do so
- P2P systems are vulnerable against **selfish** behavior
 - Missing **incentives** => free riding problem (e.g. Gnutella)
- Some P2P applications use simple **barter trade mechanisms** to **enforce balance** between contribution and consumption
 - E.g. **BitTorrent's** tit-for-tat mechanism, **eMule's** credit system
- Weaknesses
 - Barter trade only works **bilaterally** and **immediate**
 - Exchanged goods must be of **equal value**
 - Missing **valuation** mechanisms
=> **equal** goods (e.g. file parts)
 - No support for **commercial applications**
 - No possibility to pay for more consumption
or to get paid for more contribution

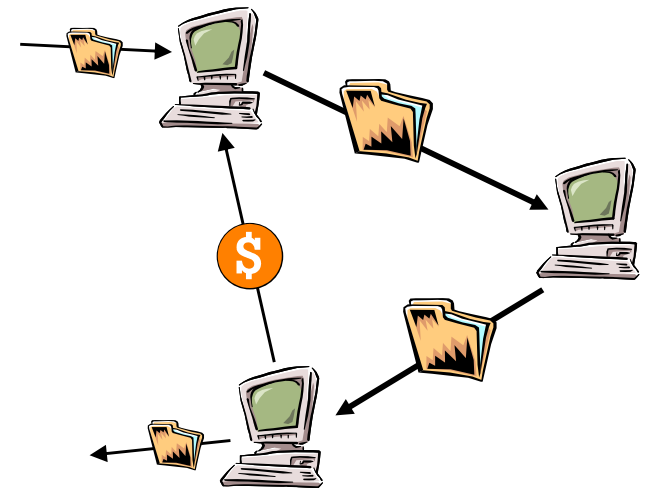


Problem Statement

- There is a lack of **reliable** market mechanisms enabling **multilateral**, **deferred**, and **commercial** trading of goods over a **decentralized** (P2P) system

- Needed mechanisms
 - A **pricing/valuation mechanism**
 - Taking into account the **value** of the different goods offered
 - An **accounting mechanism**
 - Aggregating and keeping track of a peer's **balance** over **several transactions**
 - Enabling **multilateral** and **deferred** trades
 - A **charging/payment mechanism**
 - Enabling **settlement** between „unequal“ peers

- Peers may act **malicious**
 - E.g. False-reporting, collusion, cheating, forgery, denial of service



Goals and Requirements

□ Goal

- To develop a **pricing** and an **accounting/charging mechanism** that
 - Provide appropriate **incentives** for peers to share **valuable** services
 - Improve **market support** and **accountability** in **P2P networks**
 - Support applications **going beyond pure file sharing**
 - Enable **commercial** and **non-commercial** applications

□ Requirements

- To find the right trade-offs between
 - **Efficiency** (by maximizing the utility of used resources)
 - **Scalability** (by avoiding central components whenever possible)
 - **Reliability** (by taking into account **malicious behavior** of peers)

=> Counter weaknesses, while keeping P2P benefits



Methodology

- Identify **current weaknesses**
- Derive **required mechanisms** to counter weaknesses
- Analyze **design space** for new mechanisms
- Select most promising **approach**
- Design approach based on a **decentralized architecture**
- Gain first **analytical results** based on the design
- Simulate the **dynamics** of the approach, adjust design
- Implement prototype on top of a **real infrastructure**
- Gain further results from **experiments** with the prototype
- Run prototype on broad scale in a **real environment**
- Compare results and draw a **conclusion**



Selection of Related Work

□ Free Riding / Incentive Problem

- E. Adar, B. Huberman: *Free Riding on Gnutella*; First Monday, October 2000.
- P. Golle, K. Leyton-Brown, I. Mironov, M. Lillibridge: *Incentives for Sharing in Peer-to-Peer Networks*; EC, October 2001.
- R. Krishnan, M. Smith, Z. Tang, R. Telang: *The Impact of Free-Riding on Peer-to-Peer Networks*; HICCS, January 2004.
- K. Lai, M. Feldman, I. Stoica, J. Chuang: *Incentives for Cooperation in Peer-to-Peer Networks*; P2P Economics, June 2003.
- J. Shneidman, D. Parkes: *Rationality and Self-Interest in Peer to Peer Networks*; IPTPS, February 2003.

□ Reputation Mechanisms

- S. Kamvar, M. Schlosser, H. Garcia-Molina: *The EigenTrust Algorithm for Reputation Management in P2P Networks*; WWW, May 2003.

□ Security Aspects

- M. Castro, P. Druschel, A. Ganesh, A. Rowstron, D. Wallach: *Security for structured peer-to-peer overlay networks*; OSDI, December 2002.

□ Pricing Mechanisms

- B. Yu, M. Singh: *Incentive Mechanisms for Peer-to-Peer Systems*; AP2PC, July 2003.
- E. Ogston, S. Vassiliadis: *A Peer-to-Peer Agent Auction*; AAMAS, July 2002.
- Z. Despotovic, J. Usunier, K. Aberer: *Towards Peer-To-Peer Double Auctioning*; HICSS, January 2004.

□ Accounting Mechanisms

- V. Vishnumurthy, S. Chandrakumar, E. Sirer: *KARMA: A Secure Economic Framework for Peer-to-Peer Resource*; P2P Economics, June 2003.
- B. Yang, H. Garcia-Molina: *PPay: Micropayments for Peer-to-Peer Systems*; CCS, October 2003.
- R. Dingledine, M. Freedman, D. Molnar: *Accountability*; Peer-To-Peer: Harnessing the Power of Disruptive Technologies, March 2001.
- B. Cohen: *Incentives Build Robustness in BitTorrent*; P2P Economics, June 2003.
- N. Liebau, V. Darlagiannis, A. Mauthe, R. Steinmetz: *Token-based Accounting for P2P-Systems*; KiVS 05, February 2005.

□ Many more...



Outline

- Motivation & Problem Statement
- Goals and Requirements
- Auction-based Pricing with PeerMart
 - P2P Market Model & Main Problems
 - Design Space for Pricing in P2P
 - PeerMart's Approach
- PeerMint: Accounting for Peer-to-Peer Applications
 - Remote Accounting Concept
 - PeerMint's Approach
- Preliminary Conclusions and Future Work



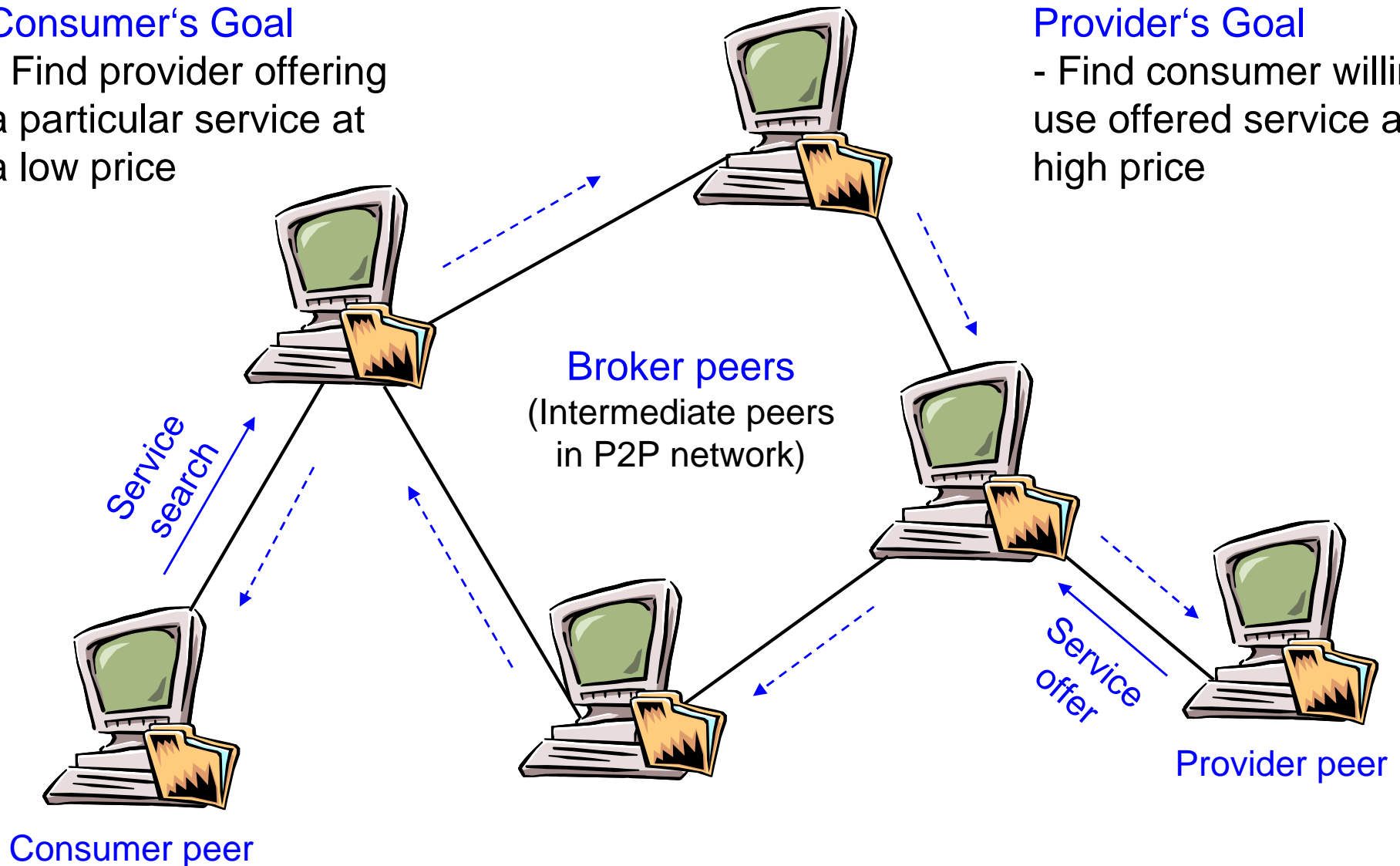
P2P Market Model: Roles and Interactions

Consumer's Goal

- Find provider offering a particular service at a low price

Provider's Goal

- Find consumer willing to use offered service at a high price



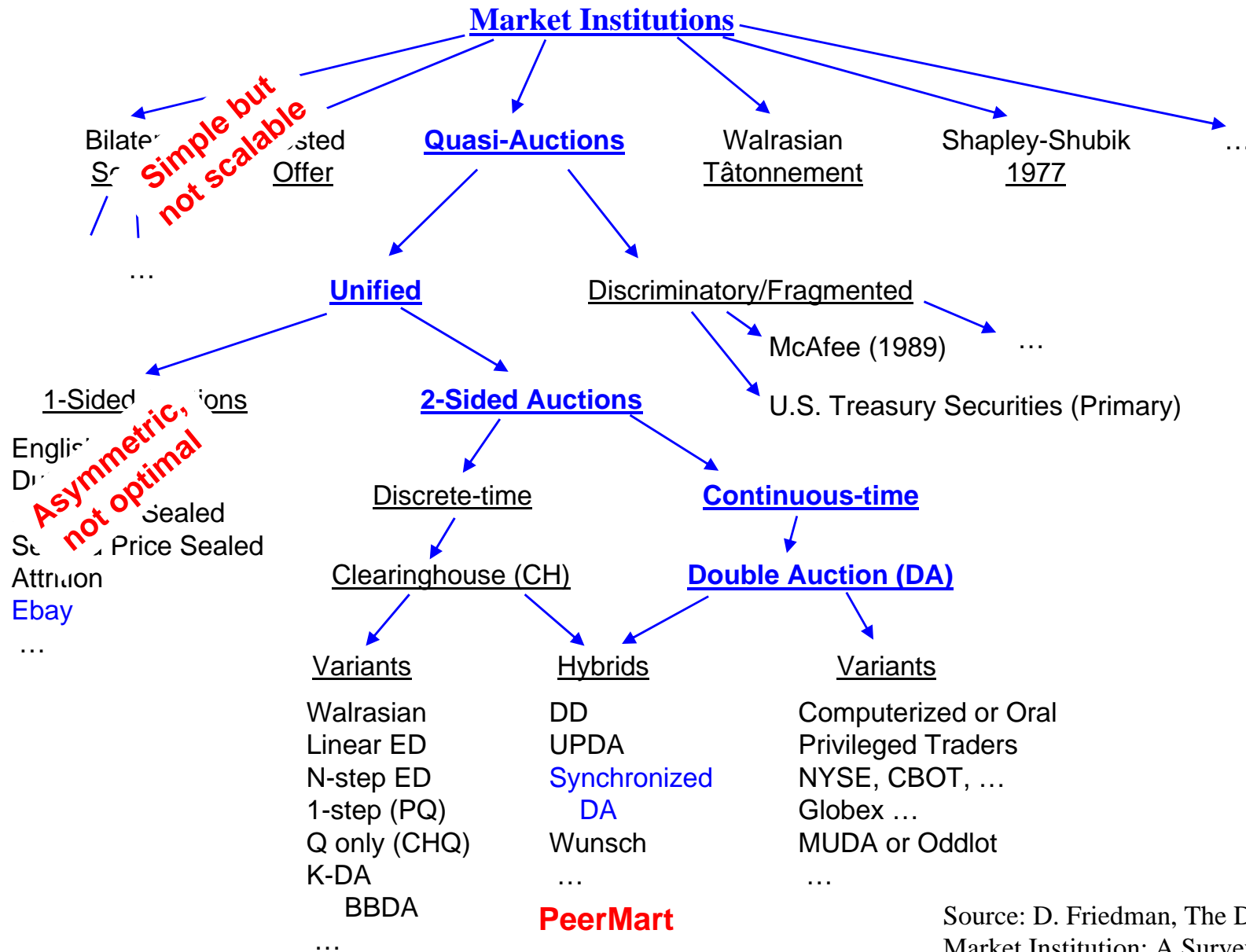
Problems and Challenges in a P2P Market

- Two different types of **free riding** should be distinguished:
 - Peers that don't provide **high-level services**
 - E.g. files, storage space, or computing power
 - Peers that don't provide **low-level services** (core functionality)
 - E.g. **forwarding**, **storing**, or **processing** information like service requests, service offers, etc.
- The use of market mechanisms is intended for **high-level services**
- For **low-level services** such mechanisms may not be feasible
 - **High technical effort** needed due to high number of small tasks
 - Low-level services difficult to **measure**
 - E.g. how to **detect** that a peer did **not forward** a message?
 - Peers might be **competitors**
 - **Provider peers** could lose opportunity to **sell** a high-level service
 - **Consumer peers** could lose opportunity to **buy** a high-level service

=> Incentive would need to be higher than potential benefit
- => Assume that a certain amount of peers behave **correctly**
 - Use **redundancy** and **reputation** to punish & reduce impact of malicious peers



Design Space for Pricing



Source: D. Friedman, The Double Auction Market Institution: A Survey, 1993



PeerMart Approach: Distributed 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 **bid price** (current highest pay price)
- Consumers **request services** they wish to use
 - Broker peers reply **ask price** (current lowest sell price)
- Brokers run the following **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)

=> Focus on **technical feasibility** of implementing a P2P auction

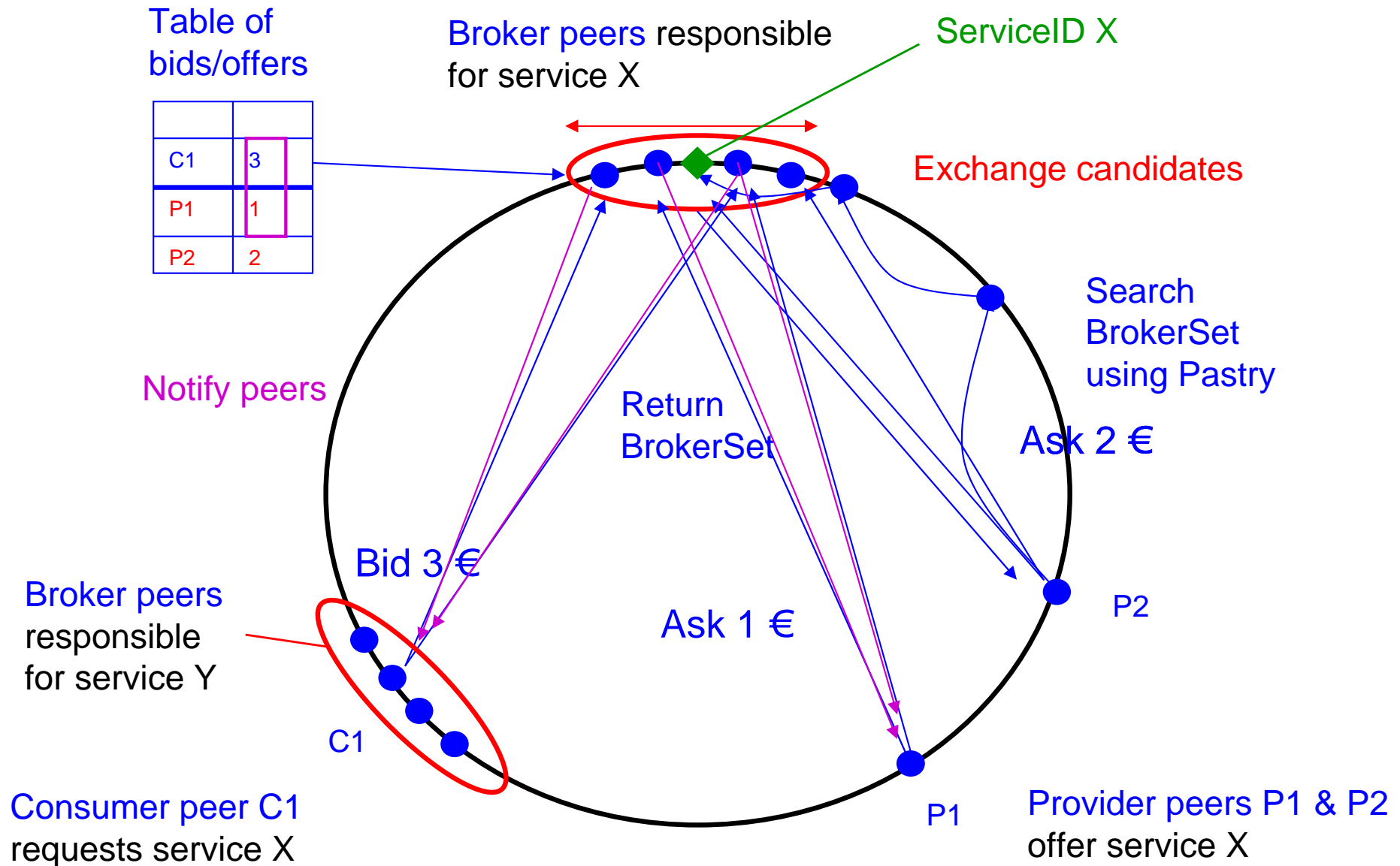


Technical Design

- Pastry is used as underlying P2P infrastructure
 - Open source implementation in Java ([FreePastry](#))
 - Unique [128-bit nodeIDs](#), n closest nodeIDs (leaf-set) stored in routing table
 - Each peer has a [public / private key pair](#) to sign messages (certified offline)
- Services are mapped onto a [redundant set of broker peers](#) (broker-set)
 - Assumption: Services have [unique IDs](#) (e.g. hash value of a file)
 - Redundancy increases reliability and trustworthiness
- Consumers/providers [randomly select](#) f out of the n broker peers
 - Load is uniformly distributed on all broker peers
- Each broker peer keeps table of [m highest bids](#) and [m 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
 - Final matches are determined using [majority decisions](#)
 - Consumers/providers are notified by the f brokers that received the offer



Double Auction in PeerMart



Analytical Results: Scalability and Efficiency

- Basic effort for **maintaining Pastry** is $O(\log_b N)$
 - **Finding** responsible **broker-set** is $O(\log_b N)$
 - All subsequent communication is **direct**
 - **Sending offers** is $O(f)$
 - **Exchange/forward offers** is $O(f \cdot n)$
 - **Notify peers** is $O(f)$
- } Only needed for matching offers } Total costs
- Avg. number of **offers stored** per peer: $s \cdot n \cdot m$
 - s : Avg. number of service involvements per peer (naturally limited)



Experiment Setup

□ Assumptions

- Each peer assigned to $s = 2$ services
- Peers follow the ZIP **bidding strategy**

– Consumer's price offer:

$$p_{bid} = \min(p_{ask} + \alpha(p_{max} - p_{ask}), p_{max})$$

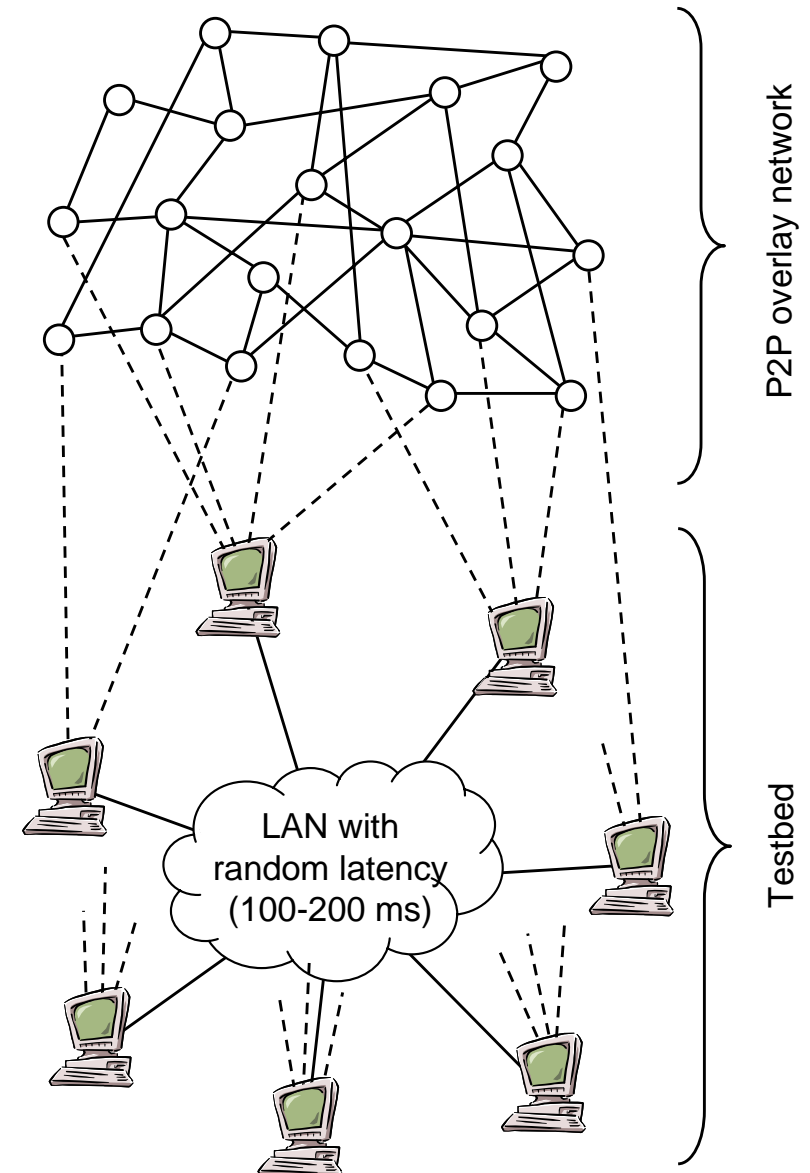
– Provider's price offer:

$$p_{ask} = \max(p_{bid} - \alpha(p_{bid} - p_{min}), p_{min})$$

– p_{max} , p_{min} are reservation prices
=> normally distributed such that
50% of offers match, $\alpha=0.1$

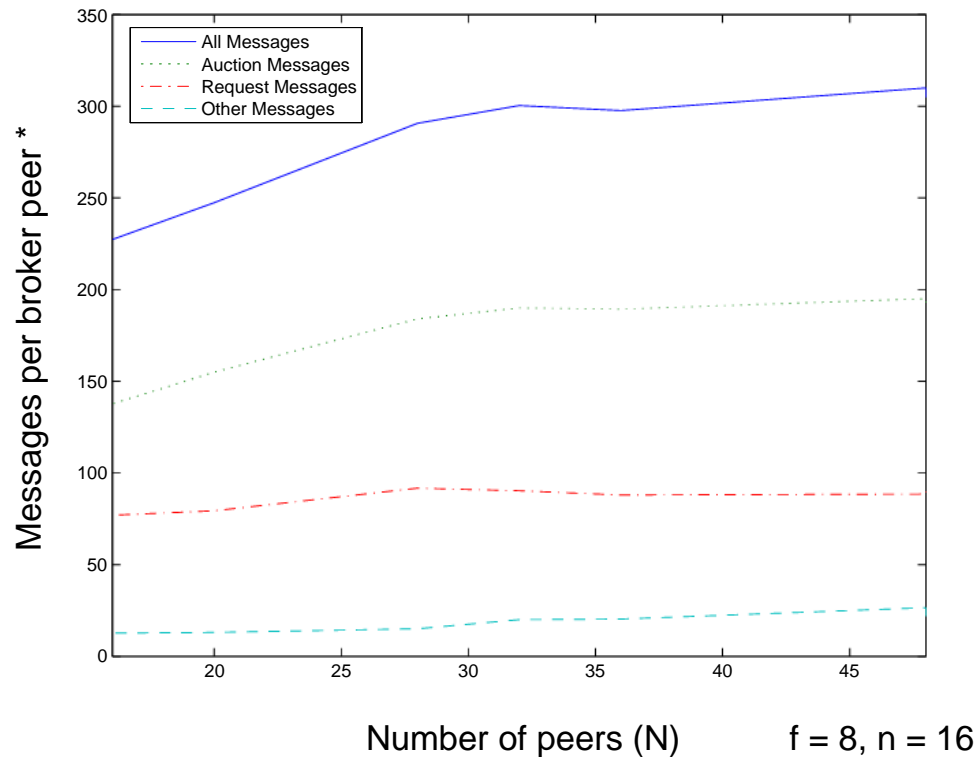
- **Malicious** peers

– Do not forward offers
nor notify peers

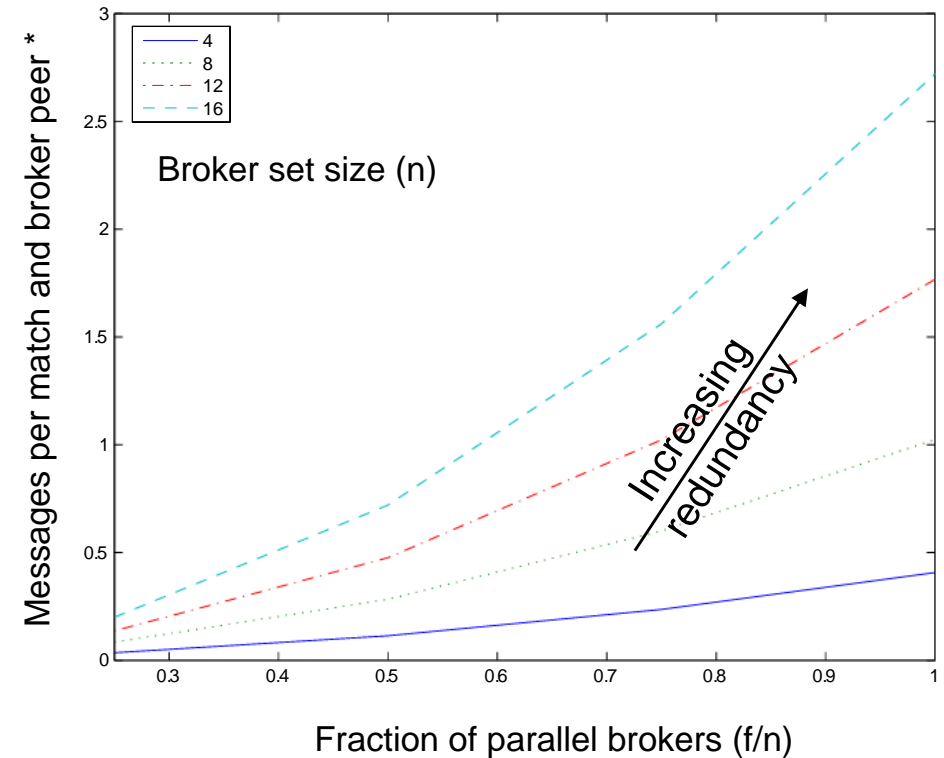


Experimental Results: Scalability, Efficiency

Scalability



Efficiency (redundancy costs)



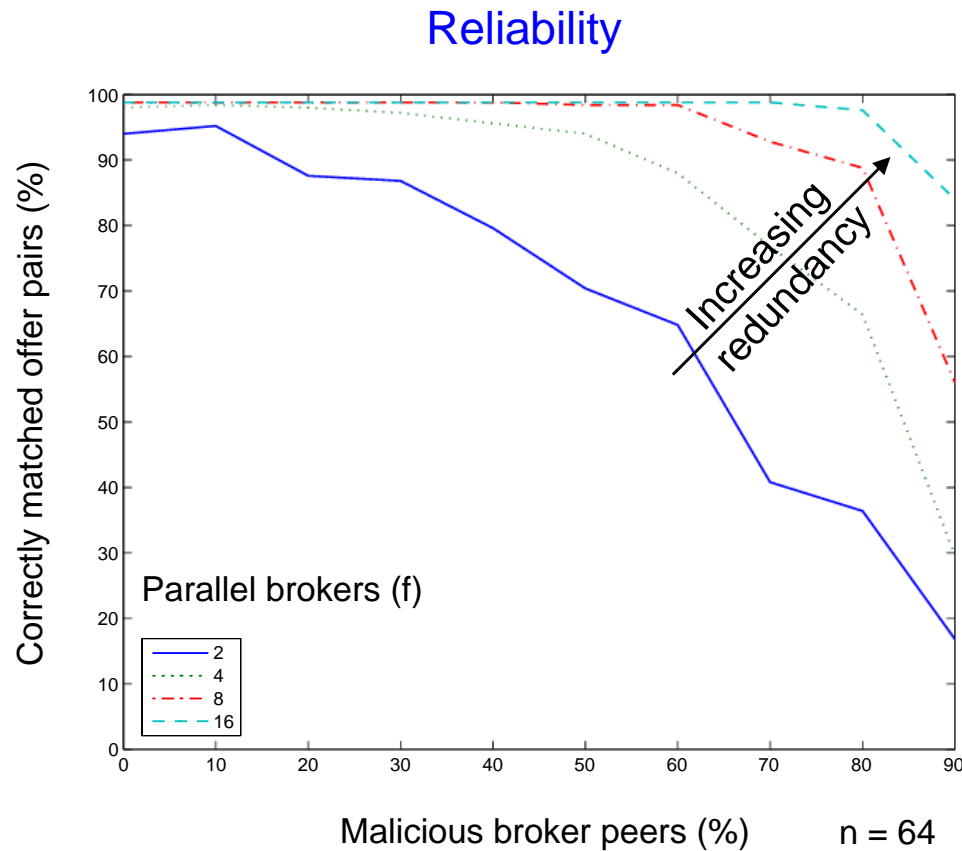
Measured security overhead:

- 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: 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



Outline

- Motivation & Problem Statement
- Goals and Requirements
- Auction-based Pricing with PeerMart
 - P2P Market Model & Main Problems
 - Design Space for Pricing in P2P
 - PeerMart's Approach
- PeerMint: Accounting for Peer-to-Peer Applications
 - Remote Accounting Concept
 - PeerMint's Approach
- Preliminary Conclusions and Future Work



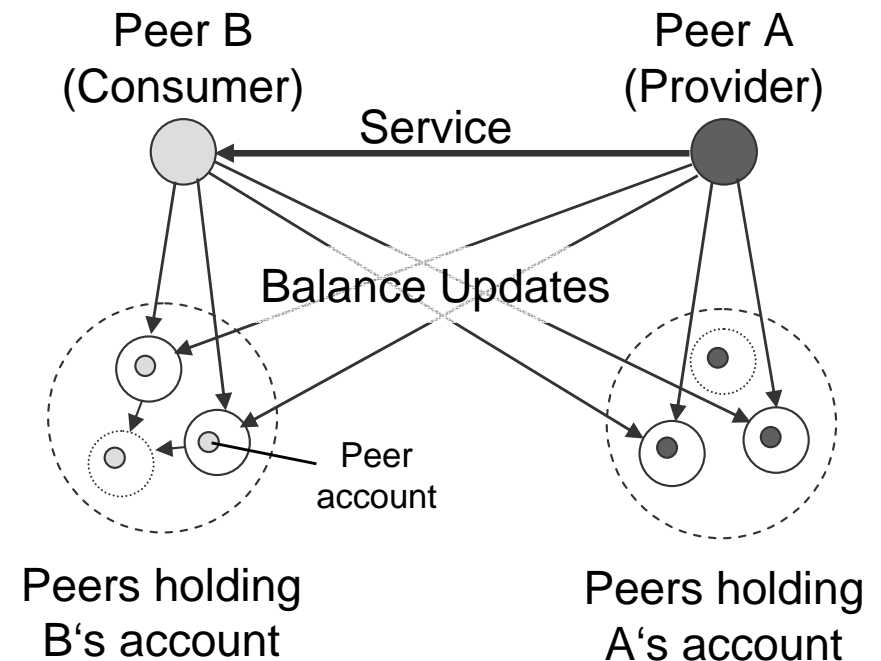
Remote Accounting Concept

□ Key Idea

- **Accounts** are **held remotely** on other peers, i.e. third party peers
- Provider / consumer send **balance updates** to remote peers
- Enables to **distribute** and **replicate** accounts over several peers
- Increases **reliability** and **availability** of the accounting data
- A higher **credibility** or **trustworthiness** can be achieved

□ Problems

- **Collusion** among peers
 - Update only provider account
- Either peer may **cheat**
 - Provider doesn't deliver service
 - Consumer sends incorrect balance update
- **Consistency** of accounts



PeerMint Approach

- Distinguishes between **peer accounts** and **session accounts**
 - **Session** = use of a **service**, e.g. file download
 - **Session accounts** keep accounting data within a session
 - **Peer accounts** aggregate information from several sessions
 - **Tariffs** specify when and by how much accounts are **updated**
- Uses a redundant set of m **session mediation peers**
 - Mediation peers keep a session account
 - **Verify** if the two peers agree and update session account accordingly
 - Update peer accounts as specified in **tariff** (e.g. at end of a session)
- Enables **charging / macro-payments**
 - A peer with a high negative credit can pay a peer with a high positive credit
 - Payments are dealt as services
- Technical Design
 - Uses **majority decisions** to overcome byzantine failures
 - Implemented on top of **Pastry** (uses keys and nodeIds from PeerMart)
 - Implements generic accounting API of **MMAPPS A&C system**



Example Session in PeerMint

□ Session Setup

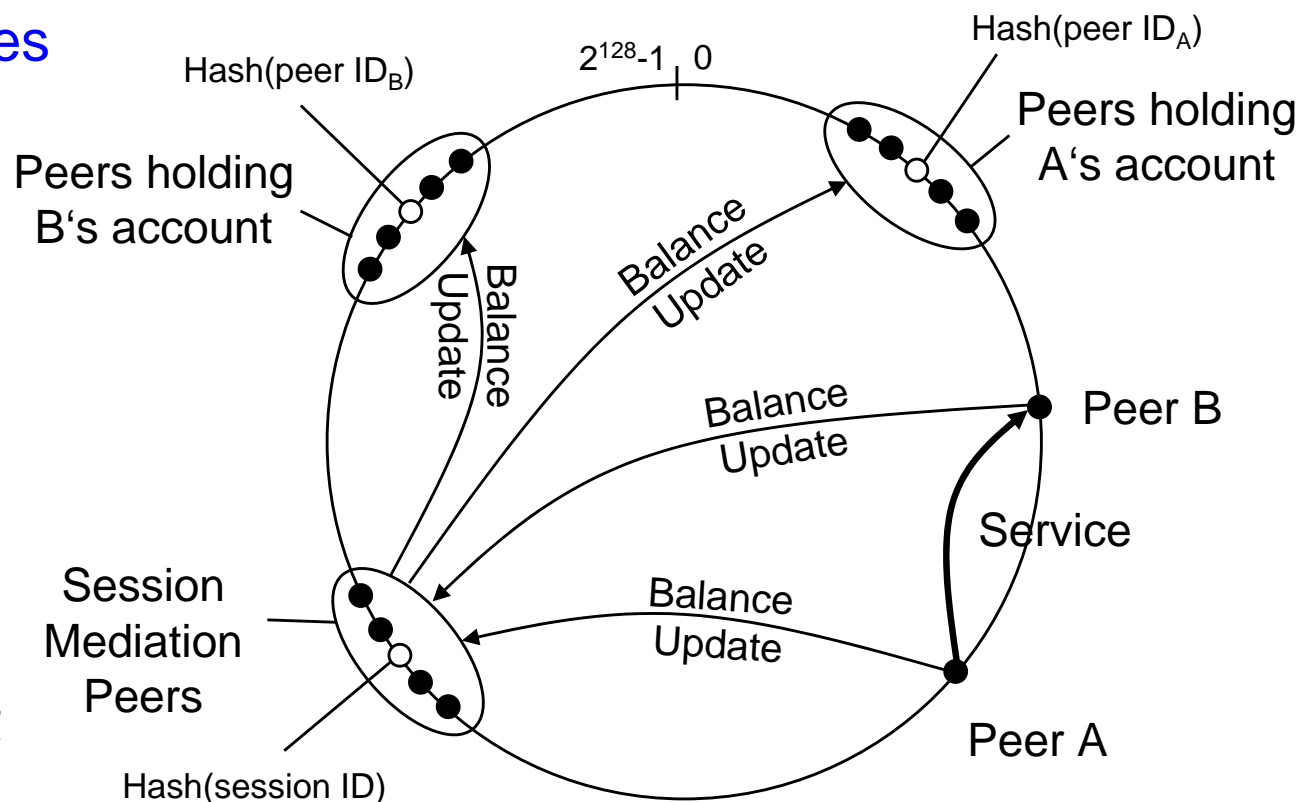
- Session partners create **signed SLA** with tariff, peerIds and sessionId
- SLA is used by session mediation peers to prove **eligibility**

□ Account Maintenance

- When **leaf-set changes** new peer becomes responsible
- Obtains current **balance** from account holders

□ Account Queries

- **No privacy**
=> any peer can query a peer account



Analytical Results: Efficiency

- Basic overhead for **maintaining Pastry overlay** is $O(\log_b N)$
 - However, PeerMint can use the infrastructure from **PeerMart**

Costs	Cost driver	Peer Accounts	Session Accounts
Message costs	Account creation	$p + O(\log_b(N))$	$m + 1 + O(\log_b(N))$
	Account update	$2mp + O(\log_b(N))$	$2m$
	Account synchronization	$p - 1$	$m - 1$
	Account query	$2p + O(\log_b(N))$	$2m$
Storage costs	Avg. #accounts per peer	p / f	ms / f

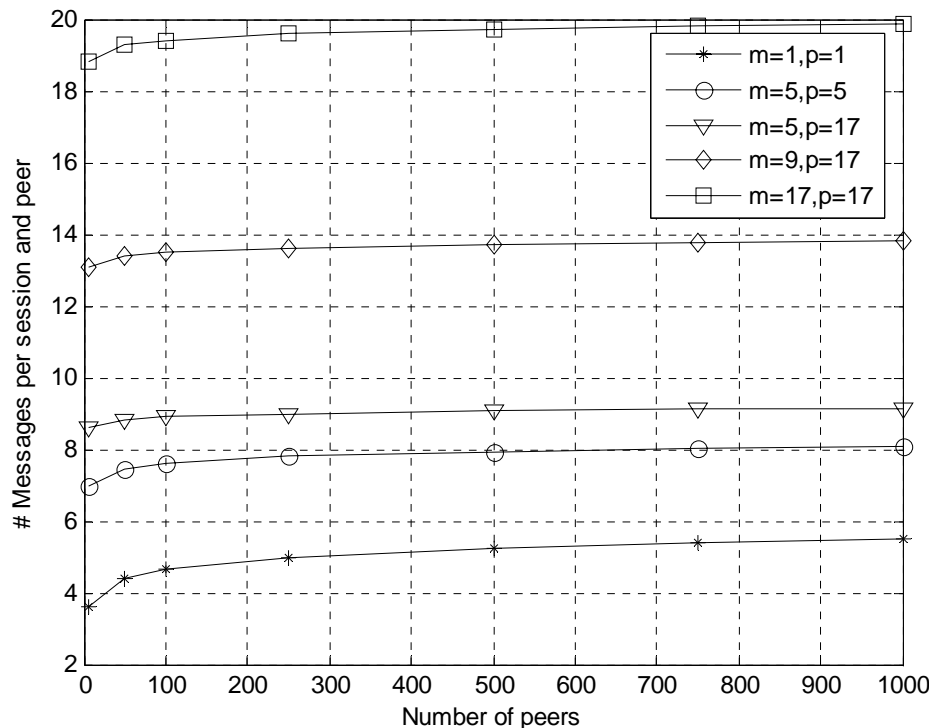
p: Number of account holders
 m: Number of session mediation peers
 n: Total number of peers
 s: Avg. number of ongoing sessions
 f: Avg. number of online peers

=> Highest effort is needed to **update peer accounts**
 However, peer account updates are less frequent than session account updates



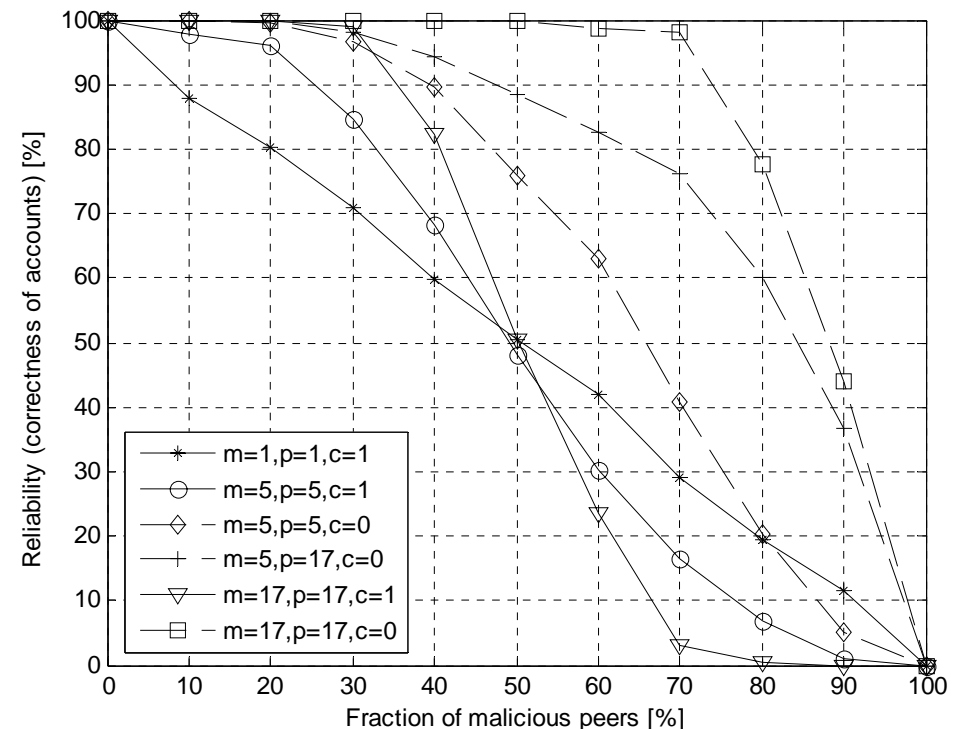
Experimental Results

Scalability, Efficiency



Malicious peers modeled as account holders reporting consistent ($c=1$) or arbitrary ($c=0$) false values (c relates to amount of collusion among malicious peers).

Reliability



- PeerMint correctly keeps accounts with **< 30% malicious colluding peers** by using **17 peer account holders** and **17 session mediators**.
- Without collusion, reliability can be achieved with **< 50% malicious peers**.



Preliminary Conclusions and Future Work

- Mechanisms are **efficient** and **scale well**
- **High reliability** even in the presence of malicious peers
 - Achieved through redundancy
- **Generic** service support through use of **tariffs**
- Prototype implemented based on **FreePastry** and embedded into generic **A&C system**
- Future Work
 - Run prototype in a real-world environment (e.g. PlanetLab)
 - Punish peers that do not stay to an offered price
=> use **Reputation for Trust**
 - Study other forms of malicious attacks, e.g. DDoS
 - Consider and prevent overlay splitting

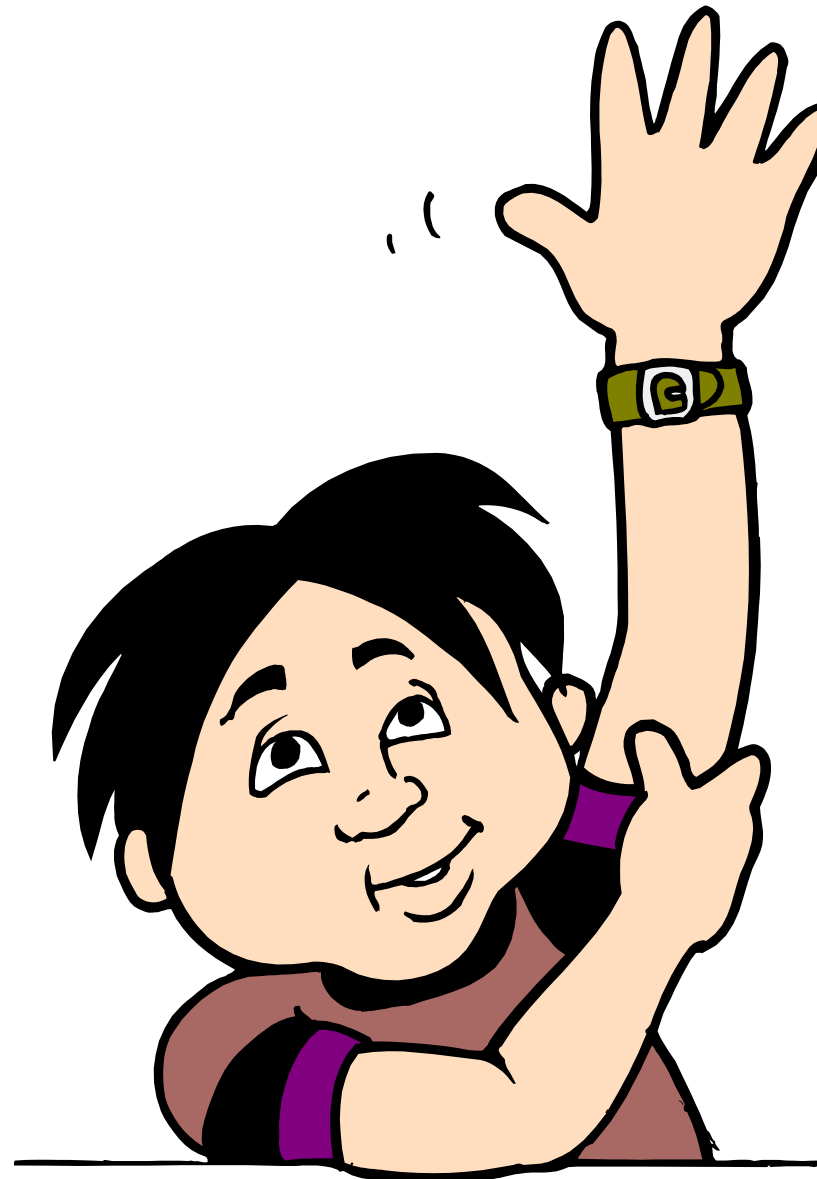


Prior Publications

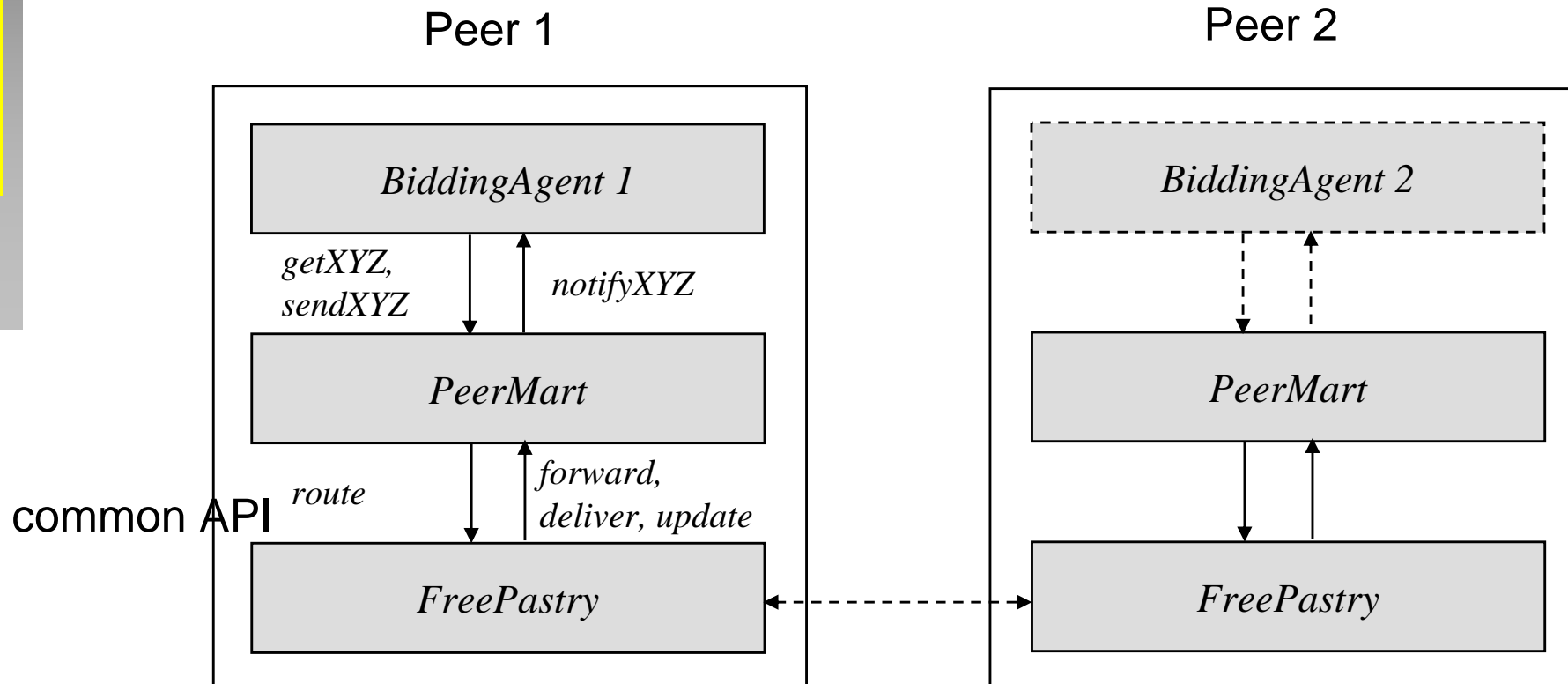
- D. Hausheer, B. Stiller: *Decentralized Auction-based Pricing with PeerMart*; 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), Nice, France, May 2005.
- D. Hausheer, B. Stiller: *PeerMart: The Technology for a Distributed Auction-based Market for Peer-to-Peer Services*; 40th IEEE International Conference on Communications (ICC 2005), Seoul, Korea, May 2005.
- D. Hausheer, B. Stiller: *PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications*; 2005 IFIP Networking Conference, University of Waterloo, Waterloo Ontario Canada, May 2005.
- D. Hausheer, J. Gerke, B. Stiller: *A Generic and Modular Accounting and Charging System for Peer-to-Peer Applications*; 14. Fachtagung Kommunikation in Verteilten Systemen 2005 (KiVS 05), Kaiserslautern, Germany, February 2005.
- D. Hausheer, N. Liebau, A. Mauthe, R. Steinmetz, B. Stiller: *Token-based Accounting and Distributed Pricing to Introduce Market Mechanisms in a Peer-to-Peer File Sharing Scenario*; 3rd IEEE International Conference on Peer-to-Peer Computing, Linköping, Sweden, pages 200-201, September 2003.
- Further info: <http://www.peermart.net/>



Thank you! – Any Questions?



Implementation of PeerMart

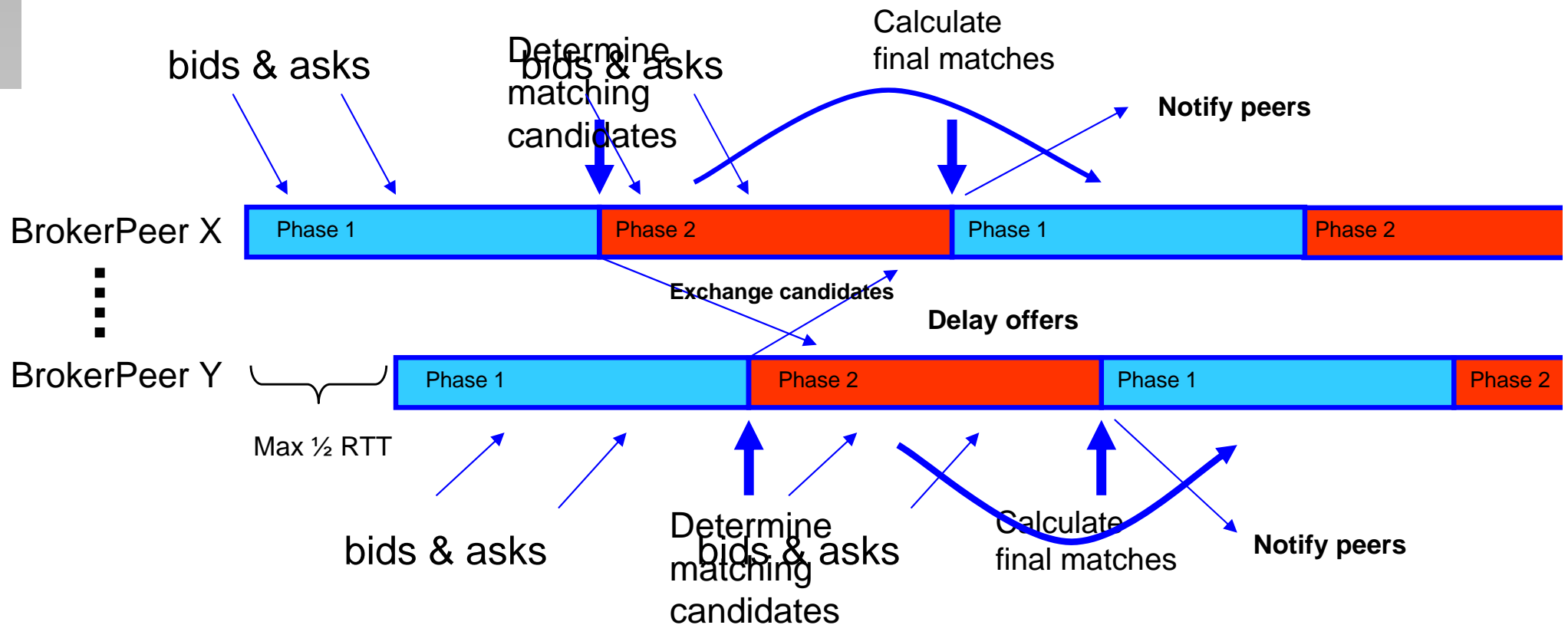


Exchange Candidates

□ Problem

- Message delay between peers
- Time is not synchronized

=> Introduce time slots greater than max Round Trip Time

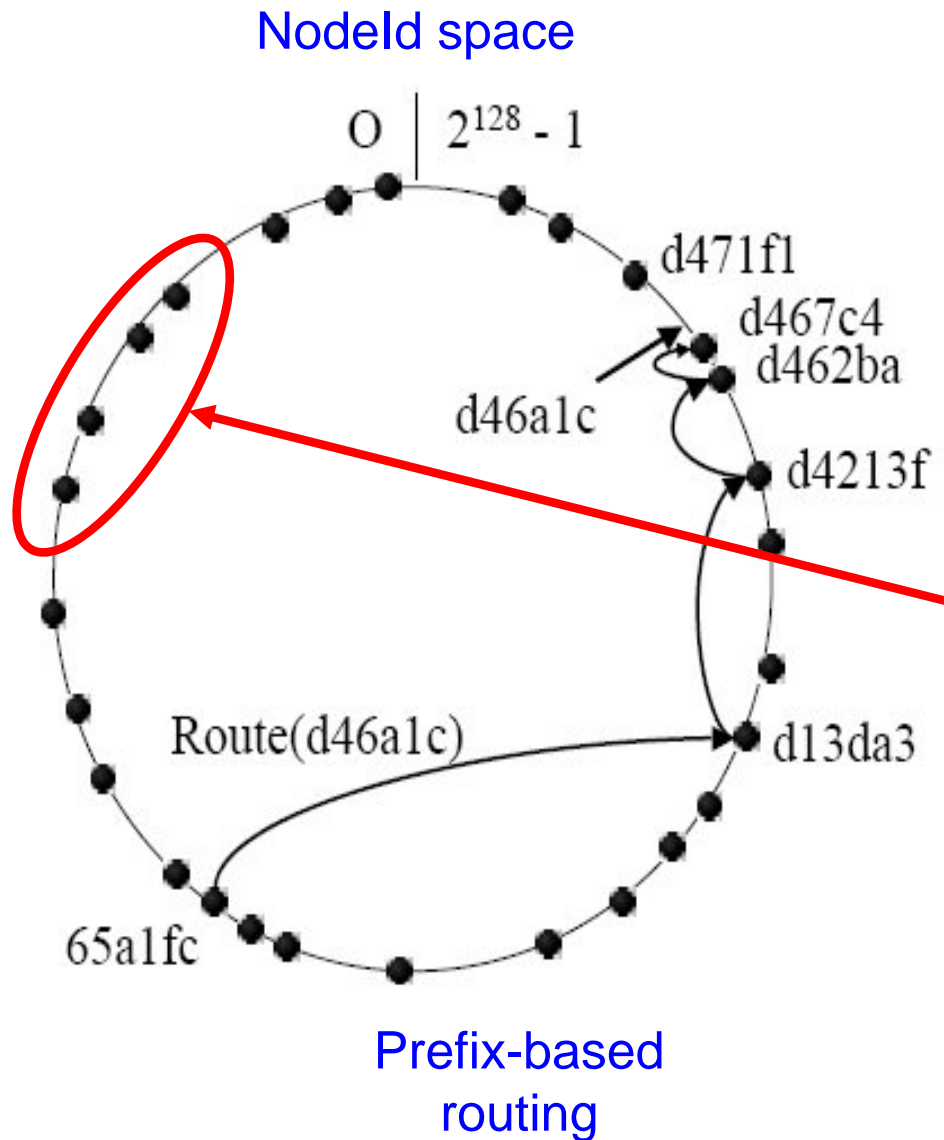


Related Work on P2P Auctions

- Z. Despotovic, J. Usunier, K. Aberer: [Towards Peer-To-Peer Double Auctioning](#); In Proceedings of the 37th HICSS Conference, January 2004.
 - Offers/Bids are broadcasted in a Gnutella-like fashion
 - Any peer can answer with a counteroffer
 - Not scalable, not strategy-proof
- E. Ogston, S. Vassiliadis: [A Peer-to-Peer Agent Auction](#); In Proceedings of AAMAS Conference, July 2002.
 - 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
- M. Fontoura, M. Ionesu, N. Minsky: [Decentralized Peer-to-Peer Auctions](#); Journal of Electronic Commerce Research, January 2005.
 - Based on Law Governed Interaction (LGI) paradigm
 - Only the auction process itself is decentralized
 - Communication overhead due to routing messages via controllers
- Many more on double auctions (not P2P) ...



Underlying Infrastructure: Pastry



Unique 128-bit **nodeids** (calculated from a peers IP address or public key using secure hash function)

Open source implementation (**FreePastry**)

Routing Table

Nodeld 10233102

Leaf set	SMALLER	LARGER
10233033	10233021	10233120 10233122
10233001	10233000	10233230 10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321



Generic and Modular Accounting & Charging System

