PeerMart: The Technology for a Distributed Auction-based Market for Peer-to-Peer Services

David Hausheer[1], Burkhard Stiller[2,1]

[1] Computer Engineering and Networks Laboratory TIK, Swiss Federal Institute of Technology ETH Zurich, Switzerland
[2] Institute for Informatics IFI, University of Zurich, Switzerland

[hausheer|stiller]@tik.ee.ethz.ch
http://www.peermart.net/

***Abstract*-P2P networks are becoming increasingly popular for a wide variety of applications going beyond pure file sharing. However, a commercial use of P2P technology is currently not possible as efficient and reliable market mechanisms are missing. This paper presents PeerMart, a distributed technology in support of a market for trading P2P services. PeerMart combines the economic efficiency of double auctions with the technical efficiency and resilience of structured P2P networks. The system is implemented on top of a redundant P2P infrastructure and is being evaluated with respect to scalability, efficiency, and reliability.**

I. Introduction

Emerging peer-to-peer (P2P) networks implement the idea that peers share resources with other peers without having to rely on a centralized infrastructure. By means of appropriate aggregation and replication techniques P2P systems can provide much higher performance and robustness than traditional client/server-based applications. Today, file sharing systems like eMule [6] or BitTorrent [4] are the most widespread P2P application. However, a number of additional applications have been developed based on P2P technology, like distributed storage, multimedia streaming, or distributed online games, which become increasingly popular.

P2P systems are based on the assumption, that every peer contributes as much as it benefits from other peers. However, as peers are autonomous entities acting in a rational and selfish way [13], it is unlikely that this kind of cooperation will happen in the absence of appropriate economic and social mechanisms. This observation is based on the well-known free-rider problem [1]. Therefore, and without appropriate incentives for peers to cooperate, P2P systems perform very badly as only few peers will offer services or resources.

To alleviate this problem, more and more P2P applications started to adopt accounting mechanisms to enforce balance between contribution and consumption of individual peers, *e.g.*, eMule's credit system [6] or BitTorrent's tit-for-tat mechanism [4]. However, these approaches are mainly file sharing-oriented and do not take into account the value of the services (files) offered. Moreover, it is not possible that credits gained for uploading a file to one peer can be spent for downloading a file from another peer or for using other services.

In order to overcome these shortcomings and enable the commercial use of P2P technology, a complete set of market mechanisms are necessary. PeerMart aims at a generic solution enabling peers to trade any kind of services with potentially different values. An example for such a service could, *e.g.*, be the upload of a file at a particular bandwidth. PeerMart provides the technology for a completely decentralized market for trading such services. In particular, it enables dynamic pricing and efficient price dissemination and lookup for services over a P2P network. Using PeerMart, peers can bid prices for services, which enables them to govern the desired service performance.

The presented approach uses an economically efficient double auction mechanism and combines it with the technically beneficial properties of a structured P2P overlay network such as Chord [15] or Pastry [12]. The core idea is to distribute the broker load of an otherwise centralized auctioneer onto clusters of peers, each being responsible for brokering several services. PeerMart uses the overlay network infrastructure to map the services onto particular sets of peers. By following this fully distributed and redundant approach, a high reliability can be achieved at a relatively low overhead of messages and necessary storage space. In addition, the solution scales very well with the number of participating peers.

The remainder of this paper is organized as follows. Section II describes the basic market model and the main problems which are focussed. In addition, an outline of the technical design space for pricing mechanisms in P2P networks is given. Section III presents the basic design and characteristics of PeerMart, while Section IV further analyzes and evaluates the system in detail. Finally, Section V concludes the paper and gives an outlook on future work. In addition, some further extensions to the basic design will be described.

II. Problem and Design Space

The considered market model for P2P services is depicted in Figure 1. For any service there are basically three different roles, consumers, providers, and intermediate peers acting as brokers. It is assumed that a peer can act in several roles at the same time. While a consumer's goal is to maximize its utility by finding providers offering a particular service at a low price, a provider's goal is to attract consumers that are willing to pay a high price to maximize its benefit for the services offered. Intermediate peers are responsible to match these needs in an efficient and optimal way, i.e. they need to forward

0-7803-8938-7/05/$20.00 (C) 2005 IEEE

messages such as service requests and service offers, as well as process and store service-related information like prices on behalf of other peers.

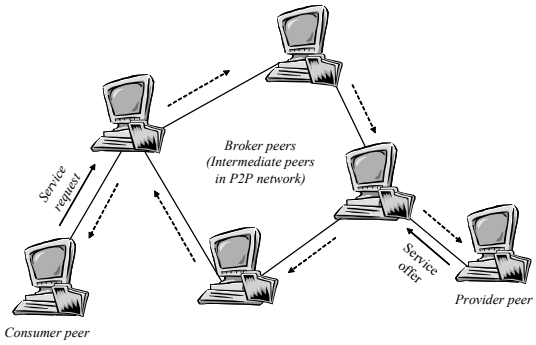

Figure 1. Market Model for P2P Services

A. Problem Statement

One of the main obstacles in a P2P environment is that peers may not be willing to cooperate or behave correctly according to a certain protocol. First of all, cooperation is costly. Therefore, peers may simply have no incentives to cooperate. But even worse, peers can be competitors for a particular service. For example, a provider could lose the opportunity to sell a service. In any case, it has to be taken into account that peers are autonomous entities which act in a rational and selfish way [13].

In general, two different types of selfish behavior can be distinguished, *a)* peers not providing services like files or other resources, and *b)* peers not providing base functionality like forwarding or caching service requests. For the first problem several solutions have been proposed in the past, such as using accounting mechanisms, micro-payment, or reputation-based schemes [7], [9], [16], [18]. However, the second problem appears to be more complicated to deal with, as this type of selfish behavior is generally much harder to detect. For example, it might be quite difficult to track down a peer which did not forward a particular message. The problem becomes even more complex when business sensitive information like price offers need to be processed or stored. Using accounting mechanisms as an incentive to provide such functionality seems not feasible due to the enormous technical effort that would be needed. Also, it might not solve the problem, since the incentive for providing that functionality would need to be higher than the potential benefit for not providing it.

PeerMart attempts to solve both problems in an integrated manner. While the answer to the first problem is to provide an efficient pricing mechanism for P2P services, the second problem is solved by using redundancy, as suggested by [14]. As it will be shown in Section IV, redundancy helps to increase the reliability of PeerMart in the presence of malicious peers.

B. Design Space for Pricing in P2P

There are many alternatives to set prices for services and disseminate them to other peers. The simplest approach would be that fixed prices are determined by a central authority, *e.g.*, the system designer. While such an approach has basically no communication overhead, it implies that the central authority has complete information to set optimal prices in advance, which seems not to be viable in practice.

In the absence of a central authority, the problem becomes technically much more complex. Prices need to be communicated to other peers in a reliable way. A simple approach would be that providers and consumers regularly broadcast price offers for services to all other peers in the network. Any offer could then directly be answered with a counteroffer by any peer. A similar approach is adopted in [5]. Obviously, such a solution does not scale and there are no guarantees whether all peers can be reached. A more scalable but rather complex approach is to store offers for later use in a form of routing table at intermediate peers. Peers could then use this information to route requests to the peer currently offering the best price rather than using broadcast. However, it is unclear how long prices should stay valid before being dropped from such a routing table.

In contrast, auction-based approaches seems to be very promising in terms of both economic and technical efficiency. While single-sided auctions have the drawback of being either provider- or consumer-oriented, double auctions enable both providers and consumers to offer prices. These offers are continuously matched by a broker following a certain matching strategy. Today the double auction is widely used in stock markets. An attempt to implement a double auction in a P2P environment is proposed in [11]. Their algorithm is based on agents which are connected in a random P2P network, and it is considered that only one commodity good is being traded. Agents randomly join to build clusters and assign a single agent as the cluster center which keeps a map of all agents in the cluster. Thus, the maximum cluster size has to be limited which implicates that the solution does not scale well. In addition, it is assumed that messages are never lost or delayed, which seems not to be a realistic assumption for a P2P network in practice.

PeerMart has been designed with a strong focus on the technical feasibility of implementing a reliable double auction mechanism on top of a P2P network. PeerMart's approach is to distribute the broker load onto redundant sets of peers. Its main challenge is to correctly synchronize offers and get to common decisions, given that a certain number of peers acts in a faulty or malicious way. This is similar to the Byzantine Generals Problem [8]. To solve this problem, PeerMart uses public key cryptography to clearly identify the sender of an offer and determines potential matches based on majority decisions among the broker peers.

III. PeerMart Design

In the following, the design of PeerMart's auction mechanism and its implementation on top of a structured P2P overlay network are described in detail.

A. Basic Auction Mechanism

The basic auction mechanism works as follows: A provider (consumer) which is interested in trading a particular service, sends a service offer (service request) to the responsible broker, which is realized by a set of peers as described later on. The broker replies with the current bid price (ask price), which is the current highest buy price (lowest sell price) offered by a peer. Based on this information, the provider (consumer) sends a price offer to the broker, using a particular strategy which can be chosen arbitrarily (a potential bidding strategy that is close to human behavior can be found in [3]). The broker continuously runs the following matching strategy:

a) Upon every price offer received from a provider (consumer), there is no match if the offer is higher (lower) than the current bid price (ask price). The price offer is dropped or stored in a table for later use.

b) If there is a match, the price offer is forwarded to the peer that offered the highest buy price (lowest sell price). The resulting price for the service is set to the mean price between the matching price offers.

B. Underlying Infrastructure

PeerMart implements the described auction mechanism on top of a structured P2P overlay network. Currently, Pastry [12] is applied, but in principle any other P2P overlay infrastructure could be used. The motivation for using Pastry is primarily its Java-based implementation, FreePastry, and its notion of leaf-sets, which represent a set of peers. PeerMart uses Pastry for peers joining or leaving the system, and to find other peers in the network. In Pastry every peer is given a unique 128-bit nodeId, which can be calculated from a peer's IP address or public key using a secure hash function. In PeerMart it is assumed that every peer has a public/private key pair, which is also used to sign and verify messages. A lightweight method that could be adopted to acquire a public/private key pair without having to rely on a public key infrastructure is described in [2]. The method is based on crypto puzzles and limits the rate at which new peers can join the system.

C. System Design

It is assumed that each service being traded over PeerMart can be described by a fixed set of parameters and has a unique serviceId with the same length as a nodeId. For a file service this can, *e.g.*, be achieved by calculating the hash value of the corresponding file. The only considered parameter which can currently vary is the price. The serviceId is mapped onto the address space of the overlay network. The set of n peers (called *broker-set*) which are numerically closest to the serviceId are responsible to act as brokers for that service. It is assumed that the serviceIds are uniformly distributed, thus every peer will on average be responsible for an equal number of services.

Each broker keeps a table for every service it is responsible for. The table has a fixed size of m rows and is used to store at most $m/2$ highest buy prices and $m/2$ lowest sell prices. In addition, the following methods are offered by a broker:

getPrice returns the current bid price (ask price) for a service. If no price offers are available, zero (infinite) is returned.

sendOffer accepts a price offer for a service. It returns true, if the offer could successfully be entered into the table. It returns false, if the price is lower (higher) than the $m/2$-highest buy price ($m/2$-lowest sell price) and therefore had to be dropped.

Furthermore, every provider (consumer) offers the following callback methods:

notifyOffer is called by a broker whenever a price offer matched with another one. The corresponding offers will be removed from the table.

notifyDrop is called when an offer had to be dropped by a broker in a later round.

An example for the double auction mechanism in PeerMart is given in Figure 2. Peer offering or requesting a particular service (with serviceId x), contact the responsible broker-set to get the current bid or ask price and then continuously send their own price offers. Apart from the price, every offer contains a sequence number and a validity time and is signed by the peer's private key. For every peer only the newest offer is kept. The validity time cannot be greater than a certain timeout t. When an offer becomes invalid, it is removed from the table.

Only the first request (to identify the broker peers for a particular service) is routed over the overlay network. All subsequent messages (namely price offer) are sent "directly" over the underlying Internet.

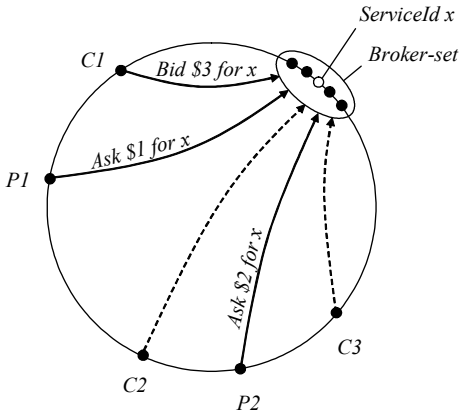

Figure 2. Double Auction in PeerMart

D. Broker-Set Maintenance

When a new service is offered for the first time, the corresponding root node (the node numerically closest to the serviceId) has to notify the other peers in its leaf-set about the new service. If the root node fails to do that (because it is a malicious node), the following fallback method is applied. Recursively, peers on the route to the serviceId are contacted, until the next closest node to the serviceId is found, which has to be in the same leaf-set as the root node. This peer then takes over the notification of the new service. In addition, every broker keeps a list of nodeIds which are in the same broker-set for a

particular service. This list is regularly updated based on changes in the leaf-set notified by Pastry.

E. Broker Redundancy

As a countermeasure against faulty or malicious peers, service offers (requests) and price offers are always sent to f randomly selected broker peers in parallel ($1 \le f \le n$). f is a design parameter that has to be set very carefully with respect to the ratio of malicious peers, desired reliability and message overhead, for which there is always a trade-off. Broker peers that receive an offer either reject it or store it in their tables according to the strategy described above. Every broker peer then forwards pairs of locally matching offers to all other peers in the broker set. Based on the signature of an offer, a broker can easily verify whether the offer is valid. If a broker has no offers that match locally, the current bid price (ask price) is forwarded instead, if it has not already been sent earlier. Based on the forwarded offers from all brokers the current bid price (ask price) is determined and a globally valid matching is performed by every broker. Whenever two offers match, the corresponding peers are notified by the brokers which initially received the offer.

In this redundant approach message loss is implicitly considered. When a message gets lost accidentally, it appears as if the corresponding peers act maliciously.

F. Timing Issues

In PeerMart slotted time is used for every individual service to tackle the problem of message delays. Every offer which potentially matches is delayed for one to two time slots before being forwarded to the other brokers. This approach guarantees that all peers have the same chance to make a deal. Every time slot has a sequence number starting at zero when a service is traded for the first time. Time slots have a fixed duration which is set to the maximum expected round-trip time in the network. During even time slots offers are collected, while during odd time slots potential matches are notified to the other broker peers as described above. Since after this synchronization process all broker peers have the same information needed to match offers, no matching conflict occurs. In the rare case that more than one peers made the same offer within the same time slot, a broker peer forwards the one that came in first. After synchronization, the offer which was forwarded by the most brokers will be selected.

IV. Evaluation

In the following, PeerMart's technical performance is evaluated with respect to efficiency, scalability, and reliability. An analysis of the economic efficiency of the proposed auction mechanism is out of the scope of this paper. Justifications for this fact can be found in [10] and [17].

A. Efficiency and Scalability

The technical efficiency of PeerMart is measured as the amount of overhead in terms of storage space used and messages generated by peers. There is a basic overhead for maintaining the Pastry overlay network. A detailed analysis of the scalability and overhead of Pastry can be found in [12].

Beyond that, PeerMart generates overhead through the exchange, storage, and matching of offers. The number of offers which need to be stored by each peer are limited in two dimensions. First, since the table size is limited, the maximum number of offers per service each broker has to maintain is m. Furthermore, it is assumed that the number of services a peer is concurrently involved in as either a consumer or provider is limited to s, as a peer's resources to consume or provide services are physically bound. Thus, on average every peer has to store a limited number of $s \cdot n \cdot m$ offers, where n is the broker-set size. Note that tables for services in which no peer is involved in anymore (no valid offers) can safely be removed.

The number of messages per service involvement a broker peer needs to deal with is influenced by several factors. Storing/rejecting an offer and notifying a match generates $2 \cdot f$ messages as only a fraction of broker peers are contacted, which will forward the message. When an offer leads to a match, an additional number of $n \cdot f$ messages are generated. A simulation has been performed with a varying number of peers bidding for services. The bidding strategy chosen for the consumers was min(*ask price* + *a**(*bid limit* - *ask price*),*bid limit*) and max(*bid price* - *a**(*bid price* - *ask limit*),*ask limit*) for the providers, respectively, where *a* is the learning parameter which has been set to 0.1. The bid and ask limits (reservation prices) of the peers were normally distributed, such that 50% of the offers were leading to a match. This bidding strategy was motivated by the ZIP strategy proposed in [3].

Figure 3 shows the message overhead per broker peer depending on the number of peers in the network. It can be

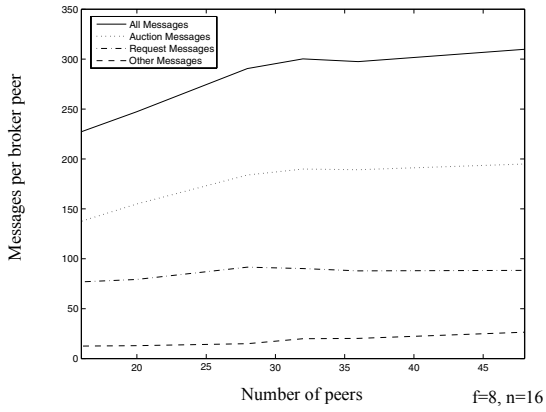

Figure 3. Message Overhead

seen that the total number of messages increases linearly in a small network, but stays almost constant when the network becomes larger. Thus, the system scales very well at a relatively low overhead of messages.

B. Reliability

Reliability denotes the capability to resist against malicious or unreliable peers. As described above, PeerMart uses redun-

dancy to achieve reliability. The redundancy of PeerMart can be configured through the parameter *f*. Thus, the desired reliability can be adjusted based on the expected number of malicious or unreliable peers. Figure 4 shows the reliability of PeerMart (measured in the amount of correctly matched offers), depending on the number of parallel brokers *f* and the amount of malicious peers in the broker set for a fixed broker set size *n*. Malicious peers were modeled as brokers which did not forward offers to other brokers. These are much harder to deal with than peers which are simply offline and can be detected after no reply has been received for a certain time. It can be seen in Figure 4, that with 8 brokers in parallel PeerMart can correctly match offers with up to 50% malicious peers. Thus, a high reliability can be achieved.

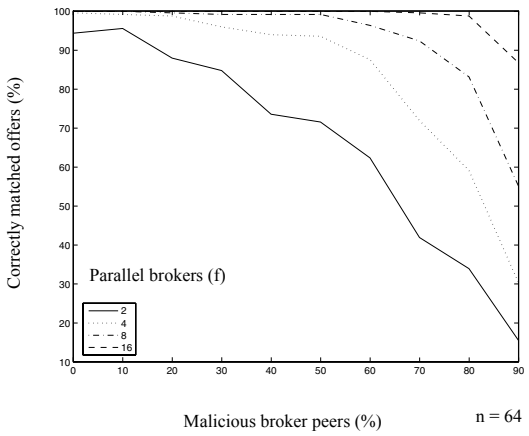

Figure 4. Reliability of PeerMart

V. Conclusions

This paper presented PeerMart, a completely distributed auction-based market based on P2P networks, which can be used by peers to efficiently trade services. Implemented on top of a redundant overlay network infrastructure, PeerMart provides a reliable system, which scales well even for a large number of peers trading services.

In future work PeerMart will be further extended and its efficiency will be optimized. While currently only the price for a service can vary, it will be investigated how several dynamic parameters could be supported.

Furthermore, the following additional types of malicious behavior will be considered:

a) A peer may continuously send offers and ignore any received match. This could be tackled by decreasing the reputation value of a peer whenever it ignores an offer and increasing it upon every accept. If the reputation value falls below a certain threshold, a peer's offers could simply be dropped, so the peer will be excluded from the system.

b) A peer may not stay to its promises, although having previously accepted an offer. This is out of control of PeerMart. Actually, the two involved peers may well further negotiate the price bilaterally, if both peers agree. PeerMart has no means to prevent this. Otherwise, the same reputation mechanism might be used as above.

c) A broker peer may not store or forward an offer. While this is generally not a problem, if at least a few peers behave correctly as shown above, it is still necessary to punish and thus prevent such behavior. A broker peer which does not store an offer cannot immediately be detected. However, if an offer is not forwarded by a broker peer, the receiver will detect it, since fewer than *f* messages are obtained. Again, the reputation value of the malicious peer could be decreased in this case.

Acknowledgement

This work has been performed partially in the framework of the EU IST project MMAPPS "Market Management of Peer-to-Peer Services" (IST-2001-34201), where the ETH Zürich has been funded by the Swiss Bundesministerium für Bildung und Wissenschaft BBW, Bern, under Grant No. 00.0275. Additionally, the authors would like to acknowledge discussions with all of their colleagues and project partners.

References

[1] E. Adar, B. Huberman: *Free Riding on Gnutella;* First Monday, Vol. 5, Nr. 10, October 2000.

[2] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach: *Security for structured peer-to-peer overlay networks*; In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02), Boston, MA, USA, December 2002.

[3] D. Cliff: *Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments*; Technical Report HPL-97-91, HP Laboratories, Bristol, England, 1997.

[4] B. Cohen: *Incentives Build Robustness in BitTorrent*; Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.

[5] Z. Despotovic, J. Usunier, K. Aberer: *Towards Peer-To-Peer Double Auctioning*; In Proceedings of the 37th Hawaii International Conference on System Sciences, Waikoloa, HI, USA, January 2004.

[6] The eMule Project: http://www.emule-project.net/.

[7] 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*; In Proceedings 3rd IEEE International Conference on Peer-to-Peer Computing, Linköping, Sweden, September 2003.

[8] L. Lamport, R. Shostak, M. Pease: *The Byzantine Generals Problem*; ACM Transactions on Programming Languages and Systems, vol. 4, pp. 382-401, July 1982.

[9] T. Moreton, A. Twigg: *Trading in Trust, Tokens, and Stamps*; Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.

[10] R. Myerson, M. Satterthwaite: *Efficient Mechanisms for Bilateral Trading*; Journal of Economic Theory, Vol. 29, pp. 265-281, 1983.

[11] E. Ogston, S. Vassiliadis: *A Peer-to-Peer Agent Auction*; In Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Bologna, Italy, July 2002.

[12] A. Rowstron and P. Druschel: *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*. In Proceedings of IFIP/ACM Middleware 2001, Heidelberg, Germany, November 2001.

[13] J. Shneidman, D. Parkes: *Rationality and Self-Interest in Peer-to-Peer Networks*; 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA, February 2003.

[14] J. Shneidman, D. Parkes: *Using Redundancy to Improve Robustness of Distributed Mechanism Implementations*; In Proceedings of 4th ACM Conference on Electronic Commerce (EC'03), San Diego, CA, USA, May 2003.

[15] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan: *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*; ACM SIGCOMM 2001, pp. 149-160, San Diego, CA, USA, August 2001.

[16] V. Vishnumurthy, S. Chandrakumar, E. G. Sirer: *KARMA : A Secure Economic Framework for Peer-to-Peer Resource*; Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.

[17] B. Wilson: *Incentive Efficiency of Double Auctions*; Econometrica, Vol. 53, pp. 1101-1115, 1985.

[18] B. Yang, H. Garcia-Molina: *PPay: Micropayments for Peer-to-Peer Systems*; ACM Conference on Computer and Communications Security (CCS '03), Washington, DC, USA, October 2003.