Auctions in mobile multihop ad-hoc networks following the marketplace communication pattern[*]

Hannes Frey, Daniel Görgen, Johannes K. Lehnert, and Peter Sturm

University of Trier
Department of Computer Science
54286 Trier, Germany
E-mail: {frey|goergen|lehnert|sturm}@syssoft.uni-trier.de

Abstract. This paper presents UbiBay, a self-organizing distributed auction system using a mobile multihop ad-hoc network as its sole communication platform. In order to substantially increase the probability that negotiating peers successfully reach an agreement, communication is focused on a static geographic area, called the marketplace. Users are not constrained to be at the marketplace physically, but are allowed to utilize other ones mobile devices located at the marketplace to let a software agent negotiate with others on their behalf. The negotiation protocols of UbiBay as well as a middleware architecture for applications based on the marketplace metaphor are described in this work.

1 Introduction

Auction systems enable the exchange of goods on the basis of supply and demand. They are central to any modern economy, e.g. in the form of stock exchanges. With the rise of the pervasive world-wide web they are also used by millions of internet users in private homes on a daily basis. The most prominent example of such an auction system is ebay. Selling and buying goods at these online marketplaces is fairly easy. Offers are placed by a seller with a starting price and a deadline several days ahead. Interested customers vie with one another with increasing bids until the deadline is reached. The customer with the highest bid wins and will buy the offered good. In order to reduce network traffic and to relieve customers from the need to continuously monitor the auction 24 hours a day, software agents at the marketplace can be instructed to bid up to a given limit automatically.

Auction systems for private users are a promising application domain for mobile multihop ad-hoc networks if they are limited to a specific geographical area, e.g. a small town, a suburb or the downtown area of a city. In these scenarios, mobile devices such as smart phones, Pocket PCs and subnotebooks with

[*] This work is funded in part by DFG, Schwerpunktprogramm SPP1140 "Basissoftware für selbstorganisierende Infrastrukturen für vernetzte mobile Systeme".

wireless communication facilities form an ad-hoc network. Successful communication with a negotiating partner several hops away is at least challenging or impossible in the end because of the dynamics in such a system and the high probability for transient loss of messages. Therefore, self-organization is a prerequisite for any successful solution to mobile multihop applications. Additionally, these systems should exploit the broadcast facility inherent to any wireless communication technique. In order to eliminate the imminent broadcast storm problem [8], various solutions are proposed such as limiting the number of hops a broadcast might take or narrowing the affected area by means of topological or geographical information.

A working solution for such applications is based on the marketplace metaphor [3]. A marketplace is a fixed geographical location where information is traded. Marketplaces should be located were high device density can be expected. Client requests or agents acting on behalf of the client travel to the marketplace by infecting promising nearby devices. This decision to infect another device within communication range is based primarily on the relative geographical positions of the device actually carrying the agent, the candidate device, and the marketplace itself. When arriving at the marketplace, the agent searches for matching peers by periodically announcing the set of requirements. These infrequent broadcasts are limited to a given perimeter around the geographical center of the marketplace. Hosting devices are changed if they are going to leave the marketplace. At a given deadline or if a sufficient number of matching peers is found, the successful agent will travel back to the coordinates of its homezone to find the initiator's device. Figure 1 depicts a simple marketplace example. The requirements of an

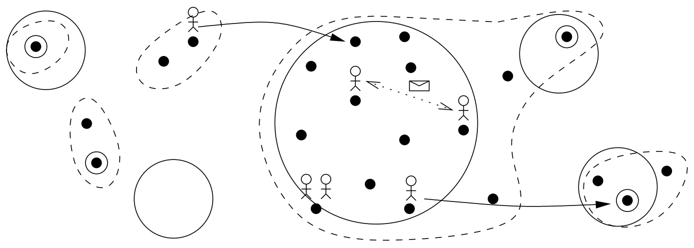


Fig. 1. A simple marketplace example with one marketplace in the middle and four user homezones. One agent tries to reach the marketplace, another one tries to reach its homezone. Two agents are matching peers and communicate with each other. The dashed lines depict the current network partitions.

auction system limited to a given geographical area match the characteristics of a self-organizing system structure with the marketplace approach as the primary communication pattern. First of all, since auctions typically last for several days, the induced latency imposed by communication between network partitions can be tolerated. Additionally, information about offered goods and the actual price should be available to any resident or visitor since these are the potential customers. Thus, the marketplace communication pattern is a perfect fit for most of the communication requirements in an auction system.

The task of securing the auction system is subject of ongoing work, but is beyond the scope of the paper. In particular, possible attacks like willful agent deletion, faked auctions or message replay will be regarded in future work. However, the auction platform is intended for low value goods, thus security is not an important issue at the moment.

In the following section `UbiBay`, an auction system for mobile ad-hoc networks using the marketplace communication pattern is described in detail. Section 3 gives a short overview of the middleware platform used to implement the presented marketplace-based `UbiBay`. In section 4 the implementation of UbiBay using a uniform workbench is described. Finally, the current and future work is pointed out.

2 UbiBay

The task of `UbiBay` is to realize a mobile auction system solely based on mobile devices with wireless communication capabilities forming a large scale multihop ad-hoc network. `UbiBay` uses the marketplace communication pattern to increase the probability that auctioneers and bidders find each others. Marketplaces are used as regions were auctions can take place. Auctions are not controlled by the users directly but by the auction agents acting on behalf of them. Users interested in auctions must retrieve information on running auctions by sending a discovery agent to the marketplace first. Thereafter they are able to send out a bid agent, to represent their biddings at the marketplace.

2.1 Auction agent

An auction at a marketplace is represented by an auction agent. It provides all relevant auction information such as product description, product category, minimum bid and auction end. It is also responsible for the actual auction process: the handling of bids, informing about outbidding and finally informing the winner of the auction.

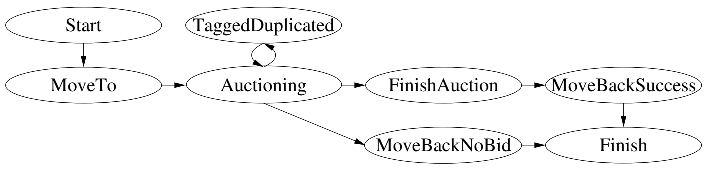


Fig. 2. Lifecycle of an auction agent

A simplified lifecycle of an auction agent is depicted in figure 2. The user starts an auction by creating an auction agent, providing it with all necessary auction information. The agent searches for an appropriate marketplace with the aid of the marketplace localization service, provided by the middleware platform. After reaching the marketplace by using the agent movement service, the agent changes to the auctioning state. In this state it is able to answer discovery requests by replying with its auction description information and to accept bids. Incoming bids with a higher value than the current bid are accepted and granted, lower or equal bids are refused. At the end of the auction, the auction agent moves to its homezone and waits for its user. After it has jumped to its user's device the agent informs the user about the result of the auction.

Since agent duplications are possible (see agent transmission protocol in section 3), all agents must take care about that. The transmission protocol ensures that the agent is informed and tagged when a duplication possibly has occured. A tagged auction agent does not accept bids immediately. After it has entered (or reentered) the marketplace it must first search for other duplicates. When it does not detect any other duplicates at the marketplace, it is allowed to switch to auctioning state. In the case that it detects an already active duplicate it deletes itself. When two or more duplicates search at the same time, one of them is elected by using the highest current device id.

It is possible that no marketplace for the `UbiBay` application exists. In that case the newly created auction agent must create a new marketplace. Applicable regions for marketplaces are provided by the agent platform. The agent chooses one, moves to this region and propagates this marketplace with aid of the marketplace localization service described in section 3.

2.2 Discovery agent

Users interested in auctions must find out about the currently running auctions. To minimize the amount of retrieved auctions and the duration of the discovery procedure, the user is able to specify the product categories, product specification and the auction related basic conditions like latest or earliest end time or the minimum (current) bid.

The agent collects information about all auctions matching its premises for a given time by asking all auction agents at the marketplace. After moving back to its homezone and jumping to its user's device it transfers all the collected information to the user. The user is now able to choose auctions to take part in.

Duplications due to message loss must not be considered due to the fact that this type of agent does not make any agreement with other agents. Nevertheless, duplicate detection can be used to reduce the amount of agents.

2.3 Bid agent

In order to participate in an auction, the user creates an agent that bids on behalf of him. He must specify his maximum bid, the height of each bid step the agent is able to increase the current bid.

The bid agent moves to the marketplace and places bids until one of its bids is accepted by the auction agent or its maximum bid is reached. If the agent's maximum bid is reached, it moves back to its homezone to inform the user that he has currently lost the auction. Otherwise, it stays at the marketplace and continues bidding if another agent places a higher bid.

The agent with the highest bid at the end of the auction is informed by the auction agent. The bid agent replies to this notification and leaves the marketplace after a short waiting time. This waiting time is needed to avoid unresolved message loss at the end of the auction. Thus, the auction agent is able to repeat the message that informs the winner until the winner replies.

Duplicated bid agents should not outbid themselves. Bid agents tagged as possibly duplicated must search for duplicates like the auction agents after entering or reentering the marketplace.

3 Marketplace platform

The `UbiBay` application uses the marketplace communication pattern to bring offers and bids together and allow self-organized auctions. Instead of directly implementing the marketplace pattern in the `UbiBay` application, a middleware platform for applications in mobile multihop ad-hoc networks is used. This middleware platform provides the application with all services needed to use the marketplace pattern.The remainder of this section will discuss the architecture and services of the middleware used by `UbiBay`.

Hardware abstraction: The foundation of the middleware is a hardware abstraction layer to unify access to basic operating system parts, to positioning, wireless communication and device discovery.

Agent movement: Agents use the agent movement service to reach specific places like marketplaces or homezones. The agent defines its target and the movement service ensures that the agent reaches its destination. When the agent arrives at the target, it is informed by the movement service.

The agent movement service transports the agent to its target by transfering the agent to other devices repeatedly. Suitable hosts are selected by a position-based greedy routing algorithm [6]. The actual transfer of the agent to the selected device is handled by the agent transport protocol. When the agent has reached its destination, the movement system informs the agent and ensures that it stays there until a new target is set.

The way back to the homezone works similar. The movement service transports the agent back to its homezone. At the homezone, the owner's device is searched using a geographically limited broadcast. When the owner's device is to far away from its homezone, it leaves a link to its current position. After the owner's device is reached, the agent is informed by the movement service.

Agent transmission protocol: Whenever the agent movement service decides to move an agent from one device to another, the agent transmission protocol is used. Due to device mobility and wireless transmission, a reliable communication channel between two devices cannot be assumed. On the other hand agents should not be lost during transmission. The agent transmission protocol solves this dilemma by tolerating agent duplication. The sender keeps a copy of the agent when sending it to another device. Both copies are marked as possibly duplicated. If all protocol messages are correctly acknowledged, the sender deletes its copy and the receiver removes the duplicate marker from its copy. Otherwise, the protocol guarantees that at least one copy of the agent exists. An agent is always marked when a duplication occurs, while the opposite does not necessarily hold.

Duplicated agents may cause problems at the marketplace when they make different agreements. Thus, upon arrival at the marketplace a duplicate elimination procedure is started for agents marked as potential duplicates

Communication at marketplaces: Due to the limited size of the marketplace, communication at marketplaces commonly involves a few hops only. The higher density of devices at the marketplace makes communication more reliable. All the communication takes place between agents not devices, since the agents might change devices during the communication. The middleware provides two kinds of communication at the marketplace: a marketplace-wide broadcast and a unicast addressing a specific agent.

The marketplace-wide broadcast is a geographically limited broadcast: messages are forwarded within the borders of the marketplace only. Therefore the marketplace broadcast causes no additional network load outside of the marketplace area. The network load inside of the marketplace area is further reduced by using a neighbor knowledge broadcast [11]. This special kind of broadcast uses only a subset of all devices to distribute a message to every device at the marketplace.

Unicasts addressing specific agents use topology-based source routing [10]. The routes commonly result from agents looking for other agents but can be explicitly built with marketplace-wide discovery broadcasts.

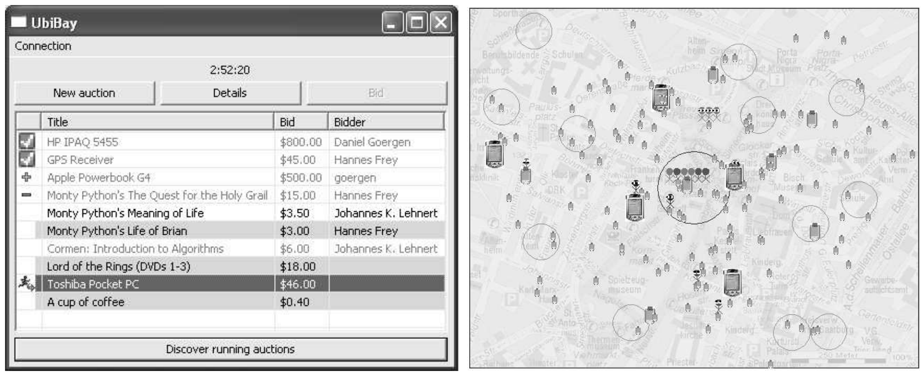
Marketplace localization: The marketplace localization service helps agents to find their corresponding marketplaces. It disseminates information on marketplace positions and applications running on them over the complete ad-hoc network. In order to avoid redundant information exchange, devices exchange hash values of their marketplace information first. Only if hash values differ, the real information is exchanged and updated.

Distributed map computation: Marketplaces should be placed in areas where a high density of mobile devices can be expected. Thus, an agent creating a new marketplace needs to know about the hotspots of the ad-hoc network, i.e. places with high device density. This is achieved by the distributed map computation service.

The area covered by the ad-hoc network is divided in small grids. Each device permanently updates the number of devices being in the grid where it is located. Additionally, it periodically broadcasts its current map state to all neighbors and listens for incoming map states. Incoming map states are averaged with locally existing states.

4 Testing UbiBay in a workbench environment

The design and implementation of UbiBay and the middleware platform follows a three-tier development principle consisting of simulation, emulation and testing in a real environment. A Java-based uniform workbench [2] especially designed for implementing applications for mobile multihop ad-hoc networks, supports this design pattern. Code reuse is the main advantage of this framework compared to other tools specialized for only one of the design steps. It turned out to be of high value to implement parts of the middleware once and test them in simulation, emulation and execution on real devices. The first version of middleware components were put into practice in the simulation part of the workbench. Repeatability of simulation experiments, fast forwarding, managing of thousands devices, and finally rich but simple visualization abstractions proved to be of high value for getting a better insight into the concept of marketplaces.



	Title	Bid	Bidder
✓	HP IPAQ 5455	$800.00	Daniel Goergen
✓	GPS Receiver	$45.00	Hannes Frey
+	Apple Powerbook G4	$500.00	goergen
−	Monty Python's The Quest for the Holy Grail	$15.00	Hannes Frey
	Monty Python's Meaning of Life	$3.50	Johannes K. Lehnert
	Monty Python's Life of Brian	$3.00	Hannes Frey
	Cormen: Introduction to Algorithms	$6.00	Johannes K. Lehnert
	Lord of the Rings (DVDs 1-3)	$18.00	
	Toshiba Pocket PC	$46.00	
	A cup of coffee	$0.40	

Fig. 3. Ubibay prototype with graphical user interface and visualization of the global simulation state.

During the design of UbiBay the problem arose when timeouts set by the agent occur during the absence from its marketplace. An early implementation used the notification mechanism of the middleware platform, in order to register as a listener for the event of leaving or reentering its designated geographic region. Thus, processing of timer events and also incoming messages is done in the context of the current agent state (inside or outside of the marketplace). It turned out that application complexity can be reduced significantly by providing a mechanism to handle this problem by the platform itself. Concerning this, the middleware platform defers (or ignores) processing of timer events and delivering of messages until the agent can be placed back to its marketplace.

Using the emulative environment allows to test UbiBay with real user interaction on real devices connected to a simulation with a high number of simulated devices. In particular, the GUI development process benefits of this approach as an intermediate step before testing the application in the real environment. But also the other application parts were improved in this development step.

Up to now only a few devices were used to test UbiBay in the real environment as a proof of concept.

5 Related work

In general, permanent network partitions are inherent to large scale ad-hoc networks, which prevents the provision of transparent end to end communication. This shows the necessity of new communication paradigms like the marketplace-based approach [3] followed by `UbiBay` and the underlying middleware platform. Communication on marketplaces does not suffer from these problems, since marketplaces define small subsets of the entire ad-hoc network.

An increasing number of middleware systems is developed specifically for mobile ad-hoc networks. Lime[9] and L^{2}imbo[1] are based on the idea of tuple-spaces, which they share between neighbored nodes. But due to the coupling of nodes, these approaches are not well-suited for highly mobile multihop ad-hoc networks. MESHMdl[4] employs the idea of tuple-spaces as well, but avoids coupling of nodes by using mobile agents, which communicate with each other using the local tuple-space of the agent platform. Proem[5] provides a peer-to-peer computing platform for mobile ad-hoc networks. STEAM[7] limits the delivery of events to geographic regions around the sender which is similar to the geographically bound communication at marketplaces. STEAM provides no long distance communication, it is only possible to receive events over a distance of a few hops.

Adaptions and extensions [10] of classical topology-based routing protocols known from static networks are coping with the problem of permanent link failures due to device mobility. However, these routing protocols will only deliver any packet, if at least sometimes there is a path from source to destination over one or more wireless links. Related to the communication pattern used by `UbiBay`, these protocols are applicable for communication inside a marketplace, since the network topology at a marketplace tends to be unpartitioned.

Another class of routing protocols is based on position information [6]. Similar to topology-based routing protocols, the objective are unpartitioned networks. Using position information is an attractive way to avoid communication overhead caused by maintaining routing information. In the marketplace communication pattern such protocols can be used to move agents within a network partition towards their destination. However, these protocols also assume that there exists a path form source to destination and have to be extented to protocols using device mobility as an additional message transport mechanism.

6 Conclusion and further work

This paper presents a working solution for auctions in mobile environments. The mobile auction system `UbiBay` uses solely mobile devices forming a large scale multihop ad-hoc network. This is achieved by a communication pattern based on the marketplace metaphor. Marketplace-based communication perfectly fits for non time critical applications in mobile multihop ad-hoc networks.

`UbiBay` and all necessary parts of the used middleware platform have been implemented both in a simulation environment and on real hardware. First tests with PocketPCs and notebooks using IEEE802.11b and GPS have been successfully carried out.

The presented components of the middleware platform are sufficient for `UbiBay`. Besides `UbiBay` also other applications like electronic ride boards and self-organized learning platforms are considered as applications using the presented middleware platform. Thus, many other features, like load balancing for marketplaces, are implemented or planned. Finally, more extensive field trials with `UbiBay` on real hardware are scheduled for the near future.

References

1. N. Davies, A. Friday, S. P. Wade, and G. S. Blair. L^{2}imbo: A distributed systems platform for mobile computing. *ACM Mobile Networks and Applications (MONET) - Special Issue on Protocols and Software Paradigms of Mobile Networks*, 3(2):143–156, August 1998.
2. Hannes Frey, Daniel Görgen, Johannes K. Lehnert, and Peter Sturm. A java-based uniform workbench for simulating and executing distributed mobile applications. In *Proceedings of FIDJI 2003 International Workshop on scientific engineering of distributed Java applications*, 2003.
3. D. Görgen, H. Frey, J.K. Lehnert, and P. Sturm. Marketplaces as communication patterns in mobile ad-hoc networks. In *Kommunikation in Verteilten Systemen (KiVS)*, 2003.
4. Klaus Herrmann. MESHMdl - A Middleware for Self-Organization in Ad hoc Networks. In *Proceedings of the 1st International Workshop on Mobile Distributed Computing (MDC'03)*, May 19 2003.
5. Gerd Kortuem. Proem: a middleware platform for mobile peer-to-peer computing. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4):62–64, 2002.
6. M. Mauve, J. Widemer, and H. Hartenstein. A survey on position-based routing in mobile ad-hoc networks. *IEEE Network Magazine*, 15(6):30–39, 2001.
7. Ren Meier and Vinny Cahill. STEAM: Event-based middleware for wireless ad hoc networks. In *22nd International Conference on Distributed Computing Systems Workshops (ICDCSW '02)*, Vienna, Austria, July 2002.
8. S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Proc. of the 5th ACM/IEEE Int. Conf. on Mobile Computing and Networking*, pages 151–162, 1999.
9. Gian Pietro Picco, Amy L. Murphy, and Gruia-Catalin Roman. LIME: Linda meets mobility. In *International Conference on Software Engineering*, pages 368–377, 1999.

10. Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, April 1999.
11. B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 194–205, 2002.