UbiBay: An auction system for mobile multihop ad-hoc networks*

Hannes Frey
University of Trier
Dept. of Computer Science
frey@syssoft.uni-trier.de

Johannes K. Lehnert
University of Trier
Dept. of Computer Science
lehnert@syssoft.uni-trier.de

Peter Sturm
University of Trier
Dept. of Computer Science
sturm@syssoft.uni-trier.de

ABSTRACT

Implementing distributed applications in mobile ad-hoc networks is a challenge because of low bandwidth, small transmission range, unpredictable topology changes and the need to conserve energy in low powered devices. This paper presents a distributed auction system using a large scale ad-hoc network as its sole communication platform. The auction system is built on top of a basic middleware service intended to be used as a generic background dissemination service for distributed self-organizing applications in mobile ad-hoc networks. This service is used to disseminate the local knowledge about the current state regarding a particular auction. The basic idea of this service is the combination of a device discovery service essential to any ad-hoc network and a dissemination service based on epidemic message distribution. This service can be used by different competing applications for permanent information dissemination, while consuming only a small fraction of the available limited network bandwidth in a mobile environment.

1. INTRODUCTION

The increasing number of todays mobile computers such as subnotebooks, PDAs and even smaller devices, their permanently improved computational power and wireless communication capabilities like Bluetooth [14] or IEEE 802.11b [1] bear the potential to serve distributed applications as an additional communication platform besides existing fixed network infrastructures. Furthermore, the investigation of the capabilities of applications solely using mobile multihop ad-hoc networks spanning over a large but limited area like a city is a challenging research field.

The very nature of large scale ad-hoc networks leads to a high probability for packet loss due to shadowing, interference and to short interaction periods in general between any two communicating devices. Furthermore, successful solutions in this area have to fulfill stringent restrictions with respect to energy consumption in order to maximize battery lifetime. The absence of a stable and dependable communication infrastructure forces these distributed applications to deploy self-organization techniques. The underlying principle of this kind of self-organization is to base all the decisions of a device on its local knowledge, to cooperate (sometimes altruistically) with immediate neighbors only, and to achieve the overall goals primarily through synergy.

Traditional middleware approaches heavily depend on a stable communication backbone and are therefore not applicable directly to this class of mobile applications. Furthermore, although it is quite clear that about the same services as traditional middleware provides are required for any self-organizing distributed application, there is still limited expertise about how to realize these services in an efficient and resource saving manner. In order to gain more insights into the efficient implementation of basic services for these systems, interesting application domains are chosen and most services – eventually to be defined as a generic middleware service – are implemented as part of the application prototype in the first place.

All these self-organizing applications build upon a generic broadcast service which implements an information dissemination protocol based on common epidemic algorithms. Ideally, this dissemination service is part of the device-specific drivers and uses periodic control messages already issued by these devices during the discovery phase to piggyback application data as broadcast information. Since the number of bits available in the control messages is severely limited, data can be distributed among subsequent control messages and the number of bits available to a given application as well as their priority is the result of a quality of service negotiation.

One of the first application domains chosen to validate and improve the generic broadcast service are auction systems for offering and selling new as well as used goods within a restricted geographical area. The goal is to find a completely decentralized and self-organizing but working solution. The broadcast service is used in this case to distribute all the offerings and bids as well as information about the trustworthiness of the participants within the user community. No further data are exchanged between any two devices using additional user-space datagram or reliable stream services. Parameters such as message frequency and the degree of data replication are crucial for the overall system performance. On the functional level, self-organizing and decentralized solutions to problems such as privacy protection, mutual suspicion between the participants, protocols to

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

guarantee fair auctions (the highest bid wins), and strategies to prohibit false bids are required.

The remainder of this paper is organized as follows. The next two sections present the UBIBAY prototype, the auction protocol and the distributed reputation scheme. In section 4 a detailed definition of the broadcast service PERIODICAST is given. The paper ends with an overview of related work in this area and a sketch of the next steps towards a real implementation of UBIBAY using mobile devices.

2. UBIBAY PROTOTYPE

Each device in the UBIBAY prototype runs an auction application which allows the user to start own auctions and to take part in other users' auctions. The complete auction application is based on three message types: auction messages, description messages and notification messages. UBIBAY messages are used to disseminate information about new auctions and new bids:

- An auction message consists of five parts: the seller's ID, who initiated the auction, the ID of the item to be sold, the actual bid, the actual bidder's ID and the time when the auction ends.
- A description message contains the description of the associated auction and may contain any information a potential buyer might need in order to decide whether to participate in an auction or not. It consists of four parts: the seller's ID, the ID of the item, the time when the auction ends and the textual description of the auction.
- Notification messages are used to notify a bidder that he won the auction. It consists of four parts: the seller's ID, the ID of the item, the winner's ID and the time when the notification ends.

As long as a device does not participate in any auction and does not start a new auction, the auction application only needs to broadcast incoming auction, description and notification messages. In order to reduce the number of unnecessarily sent messages, all devices store the highest bid they know for each auction and forward only auction messages with higher bids. Additionally, they do not disseminate messages after the end of the auction or notification period. This prevents messages from being broadcast forever.

A user starts a new auction by constructing a new auction message with its own ID, a new item ID, an undefined buyer, the minimum bid and the time when the auction should end. Additionally it puts a textual description of the auction in a new description message, together with its own ID, the item ID and the time when the auction should end. It then disseminates both messages with the help of PERIODICAST, the basic broadcast service described in section 4. Whenever it receives a new auction message with a bid for its auction, it saves this message if the bid is higher than the previously saved bid.

As soon as a device has received both auction and description message for an auction, the user may decide whether to participate or not. The user makes its own bid by replacing the current buyer's ID and bid in the auction message with its own ID and a higher bid and disseminates the changed message. A local auction agent takes care that every incoming auction message with a higher bid than the last bid is treated this way as long as the maximum bid is not reached.

At the end of an auction the seller notifies the bidder with the highest bid by disseminating a notification message. This is not done if the seller did not receive any bids. This implementation of the prototype may lead to false winners, since the seller must not have received the highest bid necessarily.

The prototype depends on the integrity of the messages, since other devices may alter incoming messages before forwarding them or suppress messages. Thus, any device could make false bids on the behalf of other users or suppress unwanted high bids from other users. Replay attacks are not a problem, since PERIODICAST is based on periodic retransmission of messages. Falsely replayed auction messages have the impact of suppressed auction messages at most, because other devices will not forward auction messages with low bids after having received auction messages with higher bids. Retransmission of notification and description messages has no impact at all.

In order to cope with altered messages, all messages need to be digitally signed by using public key cryptography. It is assumed that a central certification authority (CA) outside of the mobile ad-hoc network exists and its public key is known on all devices. When a user buys his device he obtains a public and a private key, certified by the CA. All messages that the user initially sends, are signed with his private key and contain his public key and the certificate for his public key. Any other device that receives such a signed message can check the signature because the public key of the originator is contained in the message and it can verify the authenticity of the public key by checking the certificate with the public key of the CA. Devices drop messages with bad signatures or false public keys in order to minimize the impact of manipulated messages.

Auction agents stay on the bidder's device in order to keep the highest bid a secret. Thus, even a malicious user cannot get to know the highest bid of another user. Extensions to allow auction agents to move to other devices as well as approaches to cope with message suppression are discussed in section 6. A solution for moving auction agents to other devices has to solve the problem of keeping the highest bid of the user a secret while allowing the auction agent to take part in the auction on behalf of the user.

3. REPUTATION SYSTEM

The UBIBAY prototype contains a self-organizing distributed reputation system enabling participants to estimate the trustworthiness of buyers respectively sellers regarding a certain auction. It is implemented as a standalone service, which assists users to rate or to retrieve ratings about other users. A rating might be done by the winner or the initiator of a finished aution, to publish the reliablity of each other at the sale transaction.

The current implementation of the reputation system follows the approach of Schneider et al. in [22]. The basic idea is to store a database of ratings about members of the UBIBAY community, whereby community denotes all users of

the UBIBAY prototype. The database contains both personal ratings resulting from finished auctions a member was participating in and ratings received from other members. Reputation information stored in this database is periodically distributed by the use of PERIODICAST. Thus every member's database is permanently updated and will grow over the time in order to get more reliable and complete.

Each database entry represents a reputation vector for one member. It consists of a personal opinion value v_p, the number of personal encounters with that member cnt_p, a community opinion value v_c, the number of ratings received for this member cnt_c and the last modification date of this database entry ld. The values of v_p and v_c range from -127 to 127, with -127 representing the worst and 127 the best reputation value possible.

The reputation value v_p and counter value cnt_p for one member result exclusively from local ratings on one device. Each time a new rating on that member is done on this device, cnt_p is incremented by one and v_p is computed by averaging all personal ratings including the new one.

v_c represents an average value of ratings received from other users. It is computed in a similar manner as v_p. When a personal opinion value and a community opinion value are received from another device, the new community opinion value v_c is calculated by averaging all previously received v_p and v_c values about this member including the received ones. Thereafter cnt_c is incremented by two. Thus, unlike the personal opinion value, the value of a community opinion value v_c is influenced both by personal ratings and by ratings from the community. An incoming reputation information about one member from another device is accepted only once during a given time interval, in order to avoid that one user gains to much weight. In addition an aging mechanism is realized to decrease the weight of old reputation information.

Both counter values cnt_p and cnt_c are used to calculate the average values v_p and v_c, they indicate the quality of the respective opinion values v_p and v_c. The greater a counter value the more ratings influenced the assigned opinion value.

The last modification date ld of a database entry limits the lifetime of this entries usage and is used for garbage collection. It results from the maximum over the last date the assigned member was rated on this device and all ld values received from other devices about that member.

In order to disseminate reputation information stored about members in the local database a part of this database is periodically sent using a low priority buffer of PERIODICAST. From each database entry only the values v_p, v_c and ld are transmitted. cnt_p and cnt_c are used only locally and thus are not transferred.

4. INFORMATION DISSEMINATION SERVICE

This section presents PERIODICAST, an extended device discovery protocol, providing applications with information about current adjacent neighbors. Furthermore, it allows different competing applications to use it as an additional information dissemination service. The main idea of the protocol is to piggyback application specific broadcast messages on device discovery messages already used to determine other devices in direct communication range, so that each message utilizes the full MTU size. The following subsection gives an overview of the disseminating protocol and how it is used by different competing applications. After this subsection the protocol definition by means of its event handler routines is presented. It is assumed that each event handler is processed in one atomic step. For ease of representation, obvious handlers for initialisation, deregistration and buffer deactivation are omitted. Also failure notifications for wrong buffer allocation respectively buffer activation are omitted. In the last subsection useful protocol parameters are discussed in more detail.

4.1 Overview

To use the present form of PERIODICAST as a communication platform, an application has to reserve appropriate buffer memory of fixed length and assigned priority to place its broadcast messages inside. PERIODICAST assures the adherence of certain quality of service agreements. Buffer allocation fails, if it would result in the inability to meet the service agreements of previous buffer allocations. In this case a lower service agreement has to be done or it is up to the user to shut down other applications consuming too many network resources. Broadcast messages from different competing applications on one device are scheduled in a fairly controlled manner depending on their priorities.

Every buffer has an assigned port number to enable PERIODICAST to distinguish the content of different buffers packed in one received discovery message and pass it to the applications registered at that port. To use one port for different message types, every buffer has an additional buffer id. There is no assurance that an activated buffer is sent completely in one discovery message. Thus the receiving device might get only a fragment of this buffer. To allow reassembling of received buffer fragments to the complete buffer content, each fragment it tagged with the length of the complete buffer and the position of this fragment inside the buffer. Finally, every fragment has an assigned context. Suppose the same buffer content is sent several times before it is replaced by a new content. The context value might be used as a lamport clock which is incremented after actual buffer content is replaced by new one.

4.2 Protocol definition

In order to enable PERIODICAST to pass buffer fragments of a received broadcast message and send notifications about completely handled buffers, an application has to register itself as a fragment listener respectively buffer listener for a certain buffer with the unique identifier $(port, bid)$. In this representation a process identification pid is used to pass buffer fragments and send notifications to the associated application. All receivers for buffer notifications and fragments are remembered in the sets L_b and L_f. Herewith the handlers for buffer and fragment registration are as follows.

- Process pid registered as listener for completely sent buffer $(port, bid)$:
  $L_b \leftarrow L_b \cup \{(pid, port, bid)\}$

- Process pid registered as listener for fragments from buffer $(port, bid)$:
  $L_f \leftarrow L_f \cup \{(pid, port, bid)\}$

Received broadcast messages include a unique identifier of the sending device denoted as uid'. The remainig part of the message consists of buffer fragments. These are passed to the corresponding fragment listeners by using L_f and the `pass` routine. Furthermore, the sending device and $time$, the actual time on the receiving device, are remembered in the set N of actual known adjacent devices.

- Broadcast message $(uid', frag_1, \ldots, frag_k)$ with $frag_i = (port_i, bid_i, \ldots)$ arrived:
  `if(`$\exists t : (uid', t) \in N$`)`
  $N \leftarrow N \setminus \{(uid', t)\}$
  $N \leftarrow N \cup \{(uid', time)\}$
  `forall(`$i \in \{1, \ldots, k\}$`)`
  `forall(`$(pid, port_i, bid_i) \in L_f$`)`
  `pass(`$pid, frag_i$`)`

The next event handler is called after successive time intervals of length Δs by using a garbage collection timeout event gc. The garbage collection removes all lost devices from N and sets the next garbage collection timeout by using the method `set`.

- Timeout gc for the next garbage collection occured:
  `forall(`$(uid', t) \in N$`)`
  `if(`$time - t > \Delta s$`)`
  $N \leftarrow N \setminus \{(uid', t)\}$
  `set(`$gc, \Delta s$`)`

Before an application is able to use PERIODICAST to disseminate its own data, an appropriate buffer has to be allocated for its broadcast messages. Only an unused identifier may be used for this new buffer. Unused buffers are denoted with a $length$ of 0. Buffer allocation only succeeds, if the sum of the additional buffer length and $usage_i$, the average number of bytes sent in one pass of all buffers of the given priority i, is below a fixed value l_i. An accepted buffer allocation is notified with an $alloc$ notification containing the buffer content $bytes$, which is subsequently used to put own broadcast messages inside. In order to send an allocated buffer, it has to be activated by setting its $valid$ entry to the number of bytes to be sent. Activation succeeds only if $valid$ is set to a value less or equal to the total length of the buffer. Additional, the activated buffer might be tagged with a context value ctx as motivated in section 4.1.

- Process pid requested allocation of buffer $(port, bid)$ with length $length$ and priority pri:
  `if(`$\forall i, j : port_{ij} \neq port \vee bid_{ij} \neq bid$`) {`
  $i \leftarrow pri$
  $j \leftarrow \min\{k : length_{ik} = 0\}$
  `if(`$usage_i + length \leq l_i$`) {`
  $port_{ij} \leftarrow port$
  $bid_{ij} \leftarrow bid$
  $length_{ij} \leftarrow length$
  `notify(`$alloc, pid, port, bid, bytes_{ij}$`)`
  `}`
  `}`

- Process pid activated the first $valid$ bytes of buffer $(port, bid)$ with context ctx:
  `if(`$\exists i, j : port_{ij} = port \wedge bid_{ij} = bid$`)`
  `if(`$valid \leq length_{ij}$`) {`
  $valid_{ij} \leftarrow valid$
  $ctx_{ij} \leftarrow ctx$
  `notify(`$valid, pid, port, bid$`)`
  `}`

PERIODICAST uses a timeout event bc to send its broadcast messages in subsequent time intervals of length Δt. A broadcast message contains the unique identifier of the sending device uid. The remaining part is filled with buffer fragments from one or more priorities. Buffer fragments are appended to the broadcast message as long as its length is less or equal mtu which is set to $MTU - HDR - 1$, while MTU represents the maximum transfer unit of the underlying radio network technology and HDR amounts to the number of bytes used to store the header of one fragment $(port, bid, ctx, valid, first, last)$. There are n priorities and each priority i has an assigned probability p_i to be chosen in the next broadcast message, while $p_1 > p_2 > \ldots > p_n$ and $p_1 + p_2 + \ldots + p_n = 1$. To enable fair buffer scheduling the function $perm(p_1, \ldots, p_n)$ creates a random permutation from $\{1, \ldots, n\}$ by using the probabilities $p_1, \ldots, p_n$. For each i, j the jth buffer of priority i consists of $port_{ij}$, bid_{ij}, ctx_{ij}, $length_{ij}$, $valid_{ij}$, $bytes_{ij}$ as described in previous paragraphs. Additionally, it contains a field $current_{ij}$ to store the actual position inside the buffer content PERIODICAST is working on. A buffer is completely passed when $current_{ij} \geq valid_{ij}$ holds. All listeners of a completely passed buffer are notified by a $sent$ notification. The last buffer of priority i completed by PERIODICAST is remembered in $done_i$. In order to calculate $usage_i$, the average number of bytes sent from one priority i in one complete pass of its buffers, the following two variables are used. sum_i cummulates the total number of bytes sent in the current pass of all buffers. After this, α is used to calculate the new average value $usage_i$ from the weighted sum_i value and an exponential decay of the old average values. Finally, a broadcast message is passed to the lower layer by the use of `broadcast`. After this $f(t)$ calculates the time interval for the next broadcast message transmission and a bc timeout event is set.

- Timeout bc for the next broadcast message transmission occured:
  $(\pi_1, \ldots, \pi_n) \leftarrow perm(p_1, \ldots, p_n)$
  $k \leftarrow 1$
  $m \leftarrow \langle uid \rangle$
  `while(`$k \leq n \wedge |m| \leq mtu$`) {`
  $i \leftarrow \pi_k$
  `while(`$\exists j : current_{ij} < valid_{ij}$`) {`
  `if(`$\forall j > done_i : valid_{ij} \leq current_{ij}$`) {`
  $usage_i \leftarrow (1 - \alpha) \cdot usage_i + \alpha \cdot sum_i$
  $sum_i \leftarrow 0$
  $done_i \leftarrow 0$
  `}`
  $j \leftarrow \min\{j > done_i : current_{ij} < valid_{ij}\}$
  $v \leftarrow current_{ij}$
  $w \leftarrow \min(valid_{ij} - 1, v + mtu - |m|)$

```
      m ← m · ⟨port_ij, bid_ij, ctx_ij, valid_ij, v, w,
         bytes_ij[v], bytes_ij[v + 1], ..., bytes_ij[w]⟩
      current_ij ← w + 1
      if(current_ij ≥ valid_ij) {
         forall((pid, port_ij, bid_ij) ∈ L_b)
            notify(sent, pid, port_ij, bid_ij)
         done_i ← j
      }
      sum_i ← sum_i + 1 + w − v
   }
   k ← k + 1
}
broadcast(m)
set(bc, f(Δt))
```

Note, that buffer allocations for priority i are limited to the constraint, that the sum of additionally reserved buffer length and $usage_i$ (the average number of bytes sent in one pass of all buffers with priority i) has to be less than a fixed value l_i. This QoS constraint assures, that in the average case each buffer of priority i is sent at least all $\frac{\Delta t}{p_i} \cdot \frac{l_i}{l}$ seconds, whereby l represents the payload of one broadcast message. Note furthermore, that the smaller α is chosen, the more the change of $usage_i$ will be delayed regarding buffer activation bursts of different applications or protocols using PERIODICAST as a dissemination service.

4.3 Setting the timeout intervals

The protocol definition in the previous section introduced the timeout interval Δt for the successive broadcast message transmissions, Δs as the timeout interval for the next garbage collection and a function $f(t)$ to vary the broadcast transmission timeout depending on Δt.

Short transmission timeout intervals between two successive broadcast messages will result in frequent message collisions, if PERIODICAST is based on a simple radio network technology without collision detection. Having a more sophisticated radio layer, using a network technology such as CSMA/CA, will reduce or even avoid possible message collisions, but might lead to starvation, if no fair buffer scheduling strategy is used to share the limited bandwidth on all applications willing to use the radio layer to disseminate their own data. Additionally, the shorter this interval, the higher is the energy consumption. On the other hand, if transmission timeouts are set too long, the above problems may not appear, but two successive broadcast message transmissions might be too long away from each other, so that PERIODICAST is not reasonable to be used as an information dissemination mechanism. The current implementation of PERIODICAST uses a transmission timeout interval Δt of 1 second.

A good choice of $f(t)$ is only necessary, if PERIODICAST is based on a radio network technology, where packet collisions may occur. Otherwise, $f(t)$ might be deterministic with $t \mapsto t$. In the current implementation of PERIODICAST two adjacent devices are using the same value Δt to determine the interval of two successive broadcast message transmissions. If there is a collision of the broadcast messages from these devices, $f(\Delta t)$ has to set the next transmission time in this way, that there is a small collision probability in the next broadcast message transmission. Additional, the average interval length should be Δt. Thus, if X and Y are independent identically distributed random variables resulting from $f(t)$ and Δt, the probability $\mathrm{P}\{|X - Y| < d\}$ has to be small, whereby d denotes the time used to transmit one broadcast message. While applicable for radio networks without packet collisions, it is obvious that a deterministic $f(t)$ due to $\mathrm{P}\{|X - Y| < d\} = 1$ is inapplicable when packet collisions may occur. The current implementation of PERIODICAST uses $f(t)$ to set exponential distributed transmission intervals with rate $\frac{1}{\Delta t}$. Consequently, the average interval length is Δt and the probability $\mathrm{P}\{|X - Y| < d\}$ results to $1 - e^{-\frac{d}{\Delta t}}$.

Since PERIODICAST currently does not use trajectory information of moving devices to determine if a mobile device is still accessible, a periodic garbage collection has to be done to remove lost devices from the set of actual adjacent devices N. This garbage collection interval has to be set to a suitable value to keep N up to date. The longer the value of this interval, the more devices unaccessible remain in N. On the other hand a value too short results in a permanent removal of devices still within reach. Since $f(t)$ schedules the next broadcast transmission according to an exponential distributed value with rate $\frac{1}{\Delta t}$, the next garbage collection timeout might be set as follows. Let q denote the probability that a device is removed from N although it is still in reach. Due to the memoryless property of the exponential distributed broadcast transmission timeouts, q corresponds to $e^{-\frac{t}{\Delta t}}$. Thus, PERIODICAST uses $\Delta s = -\Delta t \cdot \ln(q)$ and a fixed small value $q = 0.05$ to determine the next garbage collection timeout interval.

5. RELATED WORK

The UBIBAY prototype contains a reputation system, which enables users to determine the trustworthiness of other participants. Reputation systems became popular through the success of centralized internet auction platforms like eBay [5], because people needed to decide whether to trust buyers and sellers [17, 21]. Distributed trust models are presented in [2] and [3]. In [15] a mobile agent based restaurant recommendation system based on reputation of the raters is described. Applications in mobile ad-hoc networks demand new self-organizing reputation systems in order to cope with the absence of a trustworthy infrastructure and the decentralized data storage. Schneider et. al. [22] present a self-organizing decentralized reputation system for mobile computers, on which the reputation system described in section 3 is based.

An information dissemination service for mobile multihop ad-hoc networks like PERIODICAST, coordinating different competing distributed applications willing to use the the underlying wireless communication technology to disseminate information to a potentially high number of destination devices, can be based on different techniques. One possible approach is to base such dissemination algorithms on an existing network routing infrastructure. Routing algorithms and in particuar clustered network architectures are well studied for ad-hoc networks [4, 6, 10, 13, 18, 19, 20]. They per-

form well if the mobility pattern of the devices is moderate. In case of high node mobility maintaining such a routing infrastructure might be intractable, since frequent topology changes will result in permanent network reorganization.

Flooding is an attractive alternative [8] in highly dynamic mobile ad-hoc networks, because it emphasizes minimal state and high reliability. Blind flooding in such mobile ad-hoc networks leads to redundant rebroadcasts, contention and packet collisions, also known as the broadcast storm problem [16]. These problems might be reduced by using the knowledge about temporary adjacent mobile devices as proposed in [12], where it is assumed that there exists a base service which maintains this kind of information.

PERIODICAST allows information dissemination without any network infrastructure. Furthermore it circumvents the problems known for flooding algorithms in radio networks by postponing the propagation of information in a fairly scheduled manner.

Instead of simply using a certification authority outside of the ad-hoc network and distribute key pairs when users buy their devices, other more sophisticated approaches can be used. A distributed certification authority using threshold secret sharing mechanisms is described in [11]. In [9] algorithms for a self-organized public-key infrastructure are presented, where users store and distribute certificates.

6. CONCLUSIONS AND FUTURE WORK

This paper describes an auction system intended to be used in a large scale ad-hoc network without the aid of any other fixed network infrastructure. The presented UBIBAY application, the reputation system and the basic communication platform PERIODICAST are currently implemented in a Java based simulation environment which is used to test the functionality of the described components. A real implementation is planned by porting the existing code to a mobile computing platform consisting of PDAs equipped with IEEE 802.11b communication facilities.

The presented prototype uses digitally signed messages to avoid tampering of bids and offerings by other users. This is of importance, since each device of the ad-hoc network is used for package fowarding. Without a signature each device could potentially fake messages before forwarding them. As motivated in section 2, message replay is no serious attack, since the auction system is based on a distributed maximum computation. The attack of dropping relevant messages is of more importance and will be concerned in a future extension of this auction system. It is planned to extend the system by an observing mechanism, where each device is running in promiscuous mode, observing the messages sent from direct neighbors within communication range. Due to this observations, a device can possibly decide if another one is dropping messages resp. forwarding older messages instead of forwarding the right ones. If several devices detect such a cheating device, they might reach a consensus to isolate such a device from the ad-hoc network and to further ignore all outgoing messages from this device.

The state of an agent must be hidden from other users, since it contains a value representing the highest bid a user is willing to do. Knowledge of this value allows to fool the agent so that it continues bidding until it reaches its maximum bid. The auction system presented in this paper avoids this problem, because auction agents are only allowed to run on the device, where they where created. So the maximum bid will never be known on another device.

As a consequence of the agents remaining on the device where they were created, the whole ad-hoc network is involved in each auction, since new bids have to be disseminated through the whole network. A future more resource saving modification of the auction system will avoid flooding of the complete network in the following way. An auction is held on a certain limited geographic location, which is called a marketplace [7] in the following. Each agent who wants to participate in this auction has to move towards such a marketplace. On the market place it is allowed to broadcast his bids over the marketplace. Since the user itself is normally not located on the marketplace, the agent must be able to change his hosting device. Simply passing an agent to another device will not work, since the highest maximum bid of this agent has to be kept secure. One approach based on the concept of secret sharing is to split the information how far an agent will bid, by sharing a bit vector $b_1 b_2 \dots b_k$ on n different devices in this way, that only all n devices together can determine the the bit b_i for $1 \leq i \leq k$, while b_i is 1 if and only if the agent will continue bidding in the ith step of the auction. This assures, that these n devices can only together sucessively determine for each step of the auction if the agent will continue bidding or stop. A precise description how this secret sharing approach works will be published in subsequent work.

Distributed applications in mobile ad-hoc networks often need information about current adjacent mobile devices. Consequently there is the need for a device discovery mechanism based on the given radio network technology. If devices are moving permanently, this device discovery has to be done periodically. Disseminating information by simple flooding mechanisms leads to early network congestion. On the other hand using high level services for reliable multicast is not appropriate, when the mobility pattern is highly dynamic. The basic idea of PERIODICAST is to combine device discovery and information dissemination in one low level service using a fair buffer scheduling and appropriate discovery periods. Notice that PERIODICAST is not intended to be used solely for commuication among mobile devices in a mobile ad-hoc network. There are situations when more reliable communication and a closer coupling of mobile devices is needed. The current implementation of PERIODICAST uses a randomized schedulig strategy and fixed discovery periods. An extension of this service is planned with additional scheduling strategies and adaptive discovery periods to reduce the number of message collisions. Thus discovery periods will depend on the number of adjacent devices and their discovery periods.

REFERENCES

[1] International Standard ISO/IEC 8802-11. Information technology – telecommunications and information exchange between systems – local and metropolitan

area networks – specific requirements – part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications, 1999.

[2] A. Abdul-Rahman and S. Hailes. A distributed trust model. In *Proceedings of the New Security Paradigms Workshop (NSPW-97)*, pages 48–60, New York, September 23–26 1997. ACM.

[3] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the Hawaii International Conference on System Sciences 33*, 2000.

[4] M. Chatterjee, S.K. Das, and D. Turgut. A weight based distributed clustering algorithm for mobile ad hoc networks. *Proceedings of the 7th International Conference on High Performance Computing, LNCS 1970*, pages 511–524, 2000.

[5] eBay. The world's online marketplace. http://www.ebay.com/, 2002.

[6] M. Gerla, C. Chiang, and L. Zhang. Tree multicast strategies in mobile, multihop wireless networks. *Mobile Networks and Applications 4 (3)*, pages 193–207, 1999.

[7] D. Görgen, H. Frey, J.K. Lehnert, and P. Sturm. Marketplaces as communication patterns in mobile ad-hoc networks. *Kommunikation in Verteilten Systemen (KiVS) 2003*, Leipzig, Germany, 2003.

[8] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. *Proceedings of the third international workshop on discrete algorithms and methods for mobile computing and communications*, pages 64–71, 1999.

[9] J. P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2001.

[10] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353 of *The Kluwer International Series in Engeneering and Computer Science*. Kluwer Academic Publishers, 1996.

[11] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *9th International Conference on Network Protocols (ICNP'01)*, 2001.

[12] H. Lim and C. Kim. Multicast tree construction and flooding in wireless ad hoc networks. *Proc. 3rd Int. ACM workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 61–68, 2000.

[13] C.R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications, Vol. 15, No. 7*, pages 1265–1275, 1997.

[14] B.A. Miller and C. Bisdikian. *Bluetooth Revealed*. Prentice Hall, Upper Saddle River, NJ, 2000.

[15] L. Mui, P. Szolovits, and C. Ang. Collaborative sanctioning: applications in restaurant recommendations based on reputation. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 118–119, Montreal, Canada, May 2001. ACM Press.

[16] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, 1999.

[17] A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc., 2001.

[18] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *Proceedings of IEEE INFOCOM 97*, pages 1405–1413, 1997.

[19] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[20] C.E. Perkins. Ad-hoc on-demand distance vector routing. In *MILCOM '97 panel on Ad Hoc Networks*, 1997.

[21] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. *CACM*, 43(12):45–48, December 2000.

[22] J. Schneider, G. Kortuem, J. Jager, S. Fickas, and Z. Segall. Disseminating trust information in wearable communities. *2nd International Symposium on Handheld and Ubitquitous Computing*, 2000.