A generic background dissemination service for mobile ad-hoc networks

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

Abstract

This paper proposes a basic middleware service intended to be used as a generic background dissemination service for distributed self-organizing applications and protocol services in mobile ad-hoc networks. The basic idea is the combination of a device discovery service needed in ad-hoc networks and a dissemination service based on epidemic message distribution. This service is intended to be used by different competing applications and protocols for permanent information dissemination, while consuming only a small fraction of the available limited network bandwidth in a mobile environment. A prototype of this service is implemented and studied in a simulation environment. The paper concludes that information dissemination based on a background dissemination service is attractive if time constraints for the information distribution are low.

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 [8] or IEEE 802.11 [1]) and finally possible location awareness of such devices due to GPS receivers or other triangulation techniques, bear the potential to serve distributed applications as an additional communication platform besides existing fixed network infrastructures. Furthermore, in the near future there will be a number of emerging self-organizing distributed applications [3][10] based mainly on the use of mobile network infrastructures, the mobile ad-hoc networks formed by these devices.

Implementing distributed applications communicating solely over such a mobile wireless network infrastructure is a challeging task because of low bandwidth, unpredictable network topology changes, small transmission range and the need to preserve energy in low powered devices. Inherent properties of mobile wireless communication compared to communication over traditional static network infrastructures are a higher probability of packet loss due to shadowing and interference and possibly short interaction periods between mobile devices.

This paper focuses on the proposal of a basic middleware service `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. One possible approach is to base such dissemination algorithms on a given network routing infrastructure. Routing in ad-hoc networks is extensively studied [11, 2, 4, 7] and performs well if the mobility pattern of the mobile devices is moderate. If the mobility pattern is highly dynamic, maintaining such a routing infrastructure might be intractable. In this case flooding is an alternative to routing, since it emphasizes minimal state and high reliability [5]. Blind flooding in such mobile ad-hoc networks leads to redundant rebroadcasts, contention and packet collisions, also known as the broadcast storm problem [9]. These problems might be reduced by using the knowledge about temporary adjacent mobile devices as proposed in [6], where it is assumed that there exists a base service which maintains this kind of information.

The remainder of this paper is organized as follows.

Section 2 motivates the communication paradigma proposed by PeriodiCast and gives a simple example of the intended usage of this service. The following two sections 3 and 4 present the protocol definition and discuss an appropriate choice of timeout intervals needed in the current implementation of PeriodiCast. In the subsequent section 5, simulation results of a prototype implementation of PeriodiCast are presented. Finally, in section 6, the applicability of periodic broadcast message transmission for data dissemination is discussed and future extensions of PeriodiCast are considered.

2 Motivation

The PeriodiCast service is used to disseminate information within a restricted geographical region using wireless communication technologies. The underlying assumption is, that the density of mobile devices within this region is sufficiently high on average to use self-organizing principles for message distribution. A stationary backbone network is not required. Devices cooperate in message dissemination altruistically by means of epidemic algorithms where devices "infect" nearby mobile systems with information. Eventually, this leads to the receipt of the disseminated information in most mobile devices.

The dissemination protocol presented in this paper is based on the periodical transmission of information in order to keep the overall overhead with respect to the available bandwidth of the wireless communication system below 1 percent. Thus, not every new device getting within transmission range is infected immediately. Of course, this limits the number of bytes that can be transmitted successfully in a given time interval. The period of message infection is in the order of seconds with a substantial end-to-end latency in information transmission. This magnitude has been chosen to tackle the general problem with simple infection schemes namely the escalation in message complexity especially in case of high density (broadcast storms [9]). PeriodiCast is therefore limited to dissemination scenarios, where the information remains stable for a sufficiently long time period and where the long transmission latency is acceptable.

There are still many application scenarios, where the PeriodiCast service can be used successfully despite its limitations. Prominent examples are exhibitions and fairs, where - hopefully - many visitors meet within a limited area and share common interests such as time schedule about extraordinary events, locations of exhibitors, and e.g. information about the public transportation system. Another area of interest are company sites and university campuses where employees respectively students can use this service for news, alternatives at lunch time etc. Another application area has been investigated [3], where an self-organizing auction system uses PeriodiCast as its sole message transport protocol to support offers and bids of new as well as used goods within a local area. Here, the service is not only used to disseminate information about ongoing auctions but also to deliver replies (bids) back to the sender.

The PeriodiCast protocol is used primarily as a background dissemination service to an anonymous number of potential recipients. Additional transport protocols such as unicast with a single neighbor or certain multicasts can be used by the application on request. They are not part of this basic service. Ideally, PeriodiCast uses piggybacking to send application data as part of the control messages required by any wireless communication technique during the discovery phase in ad-hoc networks. Due to stringent limitations in the number of availalable bytes within such control messages, the space reserved for a given application is determined during a quality of service negotiation. The QoS parameters negotiated before any communication takes place are buffer sizes and priorities. Priorities are used to guarantee certain latency constraints. In general, the frequency to send data of a given priority is proportional to the priority value.

The first implemented version of PeriodiCast uses broadcast intervals with a fixed period. After buffer length and priority negotiation, an application using PeriodiCast fills its allocated buffers as needed. Completed buffers can then be validated, thus allowing PeriodiCast to broadcast the information in the near future. This is done on a fair scheduling basis with respect to the defined priorities. Typically, an application wants to be informed when the data has been send completely. This is done by means of events. For this purpose, the application is required to register itself as a event listener at the `PeriodiCast` service.

Depending on the buffer size, the `PeriodiCast` service may not be able to send the complete buffer information during a single broadcast message. Because of the highly dynamic nature of mobile ad-hoc networks, there is also a high probability for message loss. Both problems may slow down application progress substantially. Some applications may profit from further partitioning sending buffers into smaller self-contained fragments, if they are able to achieve progress with the receipt of single fragments only. For this purpose, `PeriodiCast` is able to deliver single fragments to a receiving application although the complete buffer has not been received yet.

The following example is a simplified representation of an epidemic algorithm using `PeriodiCast` to disseminate a text initially stored on one device. Devices are distinguished by their unique identifiers *uid*. The application on each device reserves one buffer for port *port* with buffer identification *bid* and length *length*. Addidtionally, it registers as a listener for received buffer fragments and completely sent buffers. The given text is split in n parts $text_1, \ldots, text_n$, whereby each part fits completely in the reserved buffer. Initially only the device owning the text starts the dissemination of the first text part $text_1$.

```
• Application started:
  reserve buffer $(port, bid)$ with length $length$
  register as fragment listener for $(port, bid)$
  register as buffer listener for $(port, bid)$
  $k \leftarrow 1$
  if($uid = 1$) {
    copy $text_1$ into buffer $(port, bid)$
    validate $(port, bid)$ with context 1
  }
```

After its buffer is completely sent it continues with the next one. Note, that the handler for completely sent buffers is not called until the buffer was validated for the first time.

```
• Buffer $(port, bid)$ completely sent:
  copy $text_k$ into the buffer $(port, bid)$
  validate $(port, bid)$ with context $k$
  $k \leftarrow (k + 1) \bmod n$
```

`PeriodiCast` does not assure that a buffer is completely sent in one message. Thus, a received fragment for context i might only contain a part of $text_i$. On receipt of a fragment of context i, the currently stored text part of $text_i$ is extended with the fragment content. After the text has been completely received for the first time, the device also starts the dissemination of the complete text.

```
• Fragment for $(port, bid)$ with context $i$ received:
  add received fragment to $text_i$
  if(text completely received for the first time) {
    copy $text_1$ into buffer $(port, bid)$
    validate $(port, bid)$ with context 1
  }
```

Note that this application uses the buffer context to distinguish the different text parts $text_1, \ldots, text_n$. In other application domains this context value might be used for other purposes (e.g. as a logical clock).

3 Protocol definition

This section presents the protocol definition of `PeriodiCast`. The representation is done by using its event handler routines. It is assumed that each event handler is processed in one atomic step. For ease of representation, obvious handlers for initialisation, deregistration and invalidation are omitted. Also failure notifications for wrong buffer allocation respectively buffer validation are omitted.

In order to enable `PeriodiCast` to pass buffer fragments of a received broadcast message and send notifications about completely handled buffers, an upper protocol layer or 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 protocol or 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' and its position pos' which contains the last value received by a GPS receiver. The remainig part of the message consist of buffer fragments. These are passed to the corresponding fragment listeners by using L_f and the `pass` routine. Furthermore, the sending device, its position and *time*, the actual time on the receiving device, are remembered in the set N of actual known adjacent devices.

- Broadcast message $(uid', pos', frag_1, \ldots, frag_k)$ with $frag_i = (port_i, bid_i, \ldots)$ arrived:

```
if($\exists pos'', t : (uid', pos'', t) \in N$)
  $N \leftarrow N \setminus \{(uid', pos'', t)\}$
$N \leftarrow N \cup \{(uid', pos', time)\}$
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($(uid', pos', t) \in N$)
  if($time - t > \Delta s$)
    $N \leftarrow N \setminus \{(uid', pos', t)\}$
set($gc, \Delta s$)
```

Before any protocol layer or 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 *length* of 0. Buffer allocation only succeedes, 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 validated by setting its *valid* entry to the number of bytes to be sent. Validation succeeds only if *valid* is set to a value less or equal to the total length of the buffer. Additional, the validated buffer might be tagged by a context value *ctx* as motivated in section 2.

- 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* validated 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* and its actual position *pos*. 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. Additional, 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:

```
$(\pi_1, \ldots, \pi_n) \leftarrow perm(p_1, \ldots, p_n)$
$k \leftarrow 1$
$m \leftarrow \langle uid, pos \rangle$
while($k \leq n \wedge |m| \leq mtu$) {
  $i \leftarrow \pi_k$
  while($\exists j : current_{ij} < valid_{ij}$) {
    if($\forall j > done_i : valid_{ij} \leq current_{ij}$) {
      $usage_i \leftarrow (1 - \alpha) \cdot usage_i + \alpha \cdot sum_i$
      $sum_i \leftarrow 0$
      $done_i \leftarrow 0$
    }
    $j \leftarrow \min\{j > done_i : current_{ij} < valid_{ij}\}$
    $v \leftarrow current_{ij}$
    $w \leftarrow \min(valid_{ij} - 1, v + mtu - |m|)$
    $m \leftarrow m \cdot \langle port_{ij}, bid_{ij}, ctx_{ij}, valid_{ij}, v, w,$
        $bytes_{ij}[v], bytes_{ij}[v+1], \ldots, bytes_{ij}[w] \rangle$
    $current_{ij} \leftarrow w + 1$
    if($current_{ij} \geq valid_{ij}$) {
      forall($(pid, port_{ij}, bid_{ij}) \in L_b$)
        notify($sent, pid, port_{ij}, bid_{ij}$)
      $done_i \leftarrow j$
    }
    $sum_i \leftarrow sum_i + 1 + w - v$
  }
  $k \leftarrow k + 1$
}
broadcast($m$)
set($bc, f(\Delta t)$)
```

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 validation bursts of different applications or protocols using `PeriodiCast` as a dissemination service.

4 Setting the timeout intervals

The protocol definition in the previous section introduced the timeout interval Δt for the successive broadcast message transmissions, Δs 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 protocol layers or 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 being no more accessible 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 Simulation results

In order to evaluate the applicability of `PeriodiCast` for information dissemination in the face of a potentially high number of mobile devices in a small geographic area, this service is implemented as a prototype in a simulation environment [3] tailored to the specific needs of distributed applications communicating solely over a mobile ad-hoc network.

`PeriodiCast` is used as the communication platform for a simple information dissemination algorithm representing an extension to the example in section 2. All devices reserve 10 buffers of size 10 kB each. At this, 5 buffers are set to the highest priority level 0 and the other 5 get the next lower priority 1. Initially one device starts the dissemination of 50 kB information with priority 0 and 50 kB information with priority 1. These are split in parts of 10 kB to fit into the previously reserved buffers. Each device is registered as a fragment listener for all used buffers. The buffer content is reconstructed by incrementally reassembling all received buffer fragments. As soon as a device has completely reconstructed the content of one buffer, it also starts the dissemination of this buffer.

The prototype is tested in a typical exhibition scenario with visitor groups changing their location frequently. This scenario is simulated on an area of 100 $m \times 100$ m with 10 booths. Each booth has an average size of 10 $m \times 10$ m. There is an extensive set of paths connecting all booths with each other. Visitors are equipped with mobile devices. They are

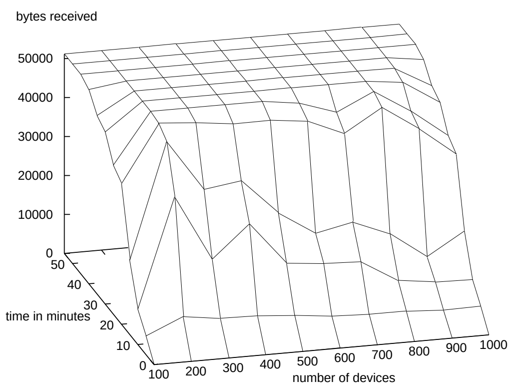

Figure 1: Average number of bytes received per device for buffers with priority 0.

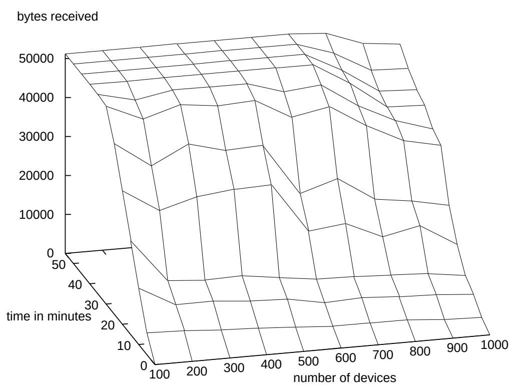

Figure 2: Average number of bytes received per device for buffers with priority 1.

combined to groups moving along the given paths with a walking speed of 0.8 meters/sec. The mobility pattern results from alternating mobility and resting phases. After such a resting phase which takes about 5 min, the group begins to move to the next randomly chosen booth. The mobile devices are equipped with a wireless communication adapter. Each transceiver has a transmission range of 10 meters, a data rate of 780 kBits/sec and a physical MTU size of 500 bytes. These values approximate typical properties of the Bluetooth technology. Each simulation run lasts approximately 1 hour in the exhibition scenario. Finally, the number of visitors is successively chosen from $\{100, 200, \ldots, 1000\}$.

With only one device starting the information dissemination, it is of particular interest how long it takes to disseminate the information in the complete network. Since a growing number of mobile devices influences the collision rate of the wireless network, the effects of higher population densities are studied. Figure 1 shows the average number of received bytes per device for buffers with priority 0 in dependence on increasing simulation time and the number of mobile devices in the simulated area. Figure 2 shows the corresponding results for buffers of priority 1.

The simulation result shows that the 50 kBytes of buffer content with the highest priority are distributed to all devices at least after 50 minutes of dissemination. The best value of 25 minutes is achieved for networks with 200 to 500 devices.

Of special interest are the results regarding the lowest and the highest simulated population density. Compared to the optimal case of 200 to 500 devices, having 100 mobile devices effects to a substantially longer period until all devices have received the desired buffer contents (45 minutes), although there are less devices willing to receive the information. This is due to the fact that there are less devices involved in the dissemination of the information. On the other hand a high number of devices might potentially speed up the dissemination process, but the more devices communicate in a small geographical area the higher the probability of message collisions will be.

Figure 2 depicts that obviously for lower priorities it takes longer to disseminate the same amount of information over `PeriodiCast`. On the other hand the shortest time it takes to disseminate information to all devices (40 minutes) is achieved by a wider range of population densities (100 to 700). Since

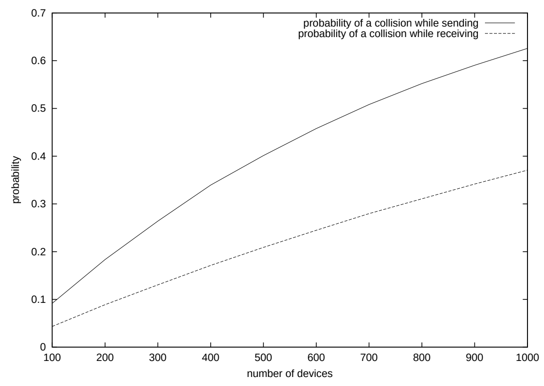

Figure 3: Probability that a sending respectively receiving device is involved in a message collision.

the number of collisions is the same for high population densities, while the number of broadcasts is lower for buffers with lower priorities, the success rate decreases for populations of 700 and more mobile devices. With the highest population density of 1000 devices, each device will receive the complete contents of the buffers with priority 1 beyond the simulated time of 55 minutes.

Finally, figure 3 shows the effect of increasing population densities on message collision probability. Both the probability that sending a packet results in a collision and the probability that a packet cannot be received due to a collision, is growing sublinearly with the population density. Thus, a high population density will decrease the number of correctly received broadcast messages per time unit, leading to the longer dissemination times observed above.

6 Conclusions and future work

In order to realize distributed applications and protocol services in mobile ad-hoc networks, a device discovery mechanism is needed. Due to unpredictable topology changes and possibly highly dynamic mobility patterns, device discovery has to be done periodically. `PeriodiCast` realizes such a distributed device discovery service in combination with a background dissemination service. Note that the additional bandwith usage of `PeriodiCast` running on a device amounts to only 0.5 percent (only 500 Bytes/sec are used of the available bandwidth of 97 kBytes/sec). Note furthermore, that `PeriodiCast` is not intended to be used as the sole communication service for mobile distributed applications. There might also be the need for a tighter coupling and a more reliable and faster communication among mobile devices.

Simulation results show that information dissemination based on `PeriodiCast` is feasible and performs well, if there are low constraints on the dissemination time. High population densities nevertheless cause problems because of high collision probabilities. Thus, since high device densities are intrinsic to the targeted application domains, `PeriodiCast` will adapt in these circumstances in a future implementation. It is planned to enhance `PeriodiCast` in a way that it increases the period of broadcast intervals in case of high device densities. Furthermore, position tracking (e.g. with the aid of a GPS device) allows the implementation of additional adaption mechanisms based on movement predictions about mobile devices.

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] 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.

[3] Ubibay: A case study for information dissemination in a distributed mobile auction system. *Submitted to Pervasive Computing 2002.*

[4] M. Gerla, C. Chiang, and L. Zhang. Tree multicast strategies in mobile, multihop wireless net-

works. *Mobile Networks and Applications 4 (3)*, pages 193–207, 1999.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] M. Papadopouli and H. Schulzrinne. Seven degrees of separation in mobile ad hoc networks. *IEEE GLOBECOM*, 2000.

[11] 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.