Efficient Capturing of Environmental Data with Mobile RFID Readers

Harald Weinschrott, Frank Dürr, and Kurt Rothermel
Institute of Parallel and Distributed Systems
Universitätsstraße 38, 70569 Stuttgart, Germany
<weinschrott|duerr|rothermel>@ipvs.uni-stuttgart.de

Abstract

In this paper we introduce a novel scenario for environmental sensing based on the combination of simple and cheap RFID-based sensors and mobile devices like mobile phones with integrated RFID readers. We envision a system that exploits the availability of these devices to cooperatively read sensors installed in the environment, and transmit the data to a server infrastructure. To achieve quality requirements and efficiency in terms of communication cost and energy consumption, this paper presents several algorithms for coordinating update operations. First, mobile nodes form an ad-hoc network for the cooperative management of requested update times to meet the desired update interval and to avoid redundant sensor reading and collisions during read operations. Second, besides this decentralized coordination algorithm, we also show a complementary algorithm that exploits infrastructure based coordination. By extensive simulations we show that our algorithms allow for autonomous operation and achieve a high quality of sensor updates where nearly 100% of the possible updates are performed. Moreover, the algorithms achieve a very high energy efficiency allowing for several hundred hours of operation assuming a typical battery of a mobile phone.

1. Introduction

City administrations need statistics to develop a city according to the behavior and the needs of its inhabitants. For example, a noise level map of a city supports decisions on traffic-reducing measures to increase the quality of living conditions. Moreover, environmental data allows for a variety of user-centric applications, e.g., a noise level or real-time air pollution map allows inhabitants to choose a good route for a walk. Urban sensing [1] allows for the dynamic generation of real-time maps and statistics of urban areas, which are fed by a large number of sensors measuring a great variety of parameters, such as noise, vibration or air pollution.

Urban sensing faces several challenges. First, large areas need to be covered with various kinds of sensors, which requires simple and cheap means of deployment and it renders the installation of special infrastructure for sensing support unfeasible. Second, without infrastructure, communication and power supply is challenging and conflicts with long lifetime of sensors. Third, coverage of large areas is only feasible with cheap sensors. Finally, defined quality of readings is needed to support applications effectively, i.e., fine-grained resolution of readings in time and space.

Two basic technologies are available today for urban sensing: sensor networks [2] and instrumented mobile devices [3]. The former can provide their readings to mobile sinks. However, sensor nodes are tailored to monitor and process environmental data autonomously. Therefore, sensor nodes provide advanced processing and communication capability, which affects cost and energy consumption. The latter can be used to collect readings directly from their surrounding. These devices provide sufficient resources, good connectivity to the infrastructure and their battery is easily rechargeable. However, for technical reasons, only a limited number of sensors can be integrated into mobile devices, e.g., a temperature sensor included in a mobile phone senses the temperature of the pocket rather than the temperature of the environment.

In this paper we propose a novel approach to urban sensing that combines the advantages of both technologies while resolving the disadvantages. We envision a system where a large number of simple and thus cheap sensors based on RFID technology [4] are installed at strategic locations in the urban environment. The functionality of these sensors is reduced to a minimum, namely sensing and one-hop communication based on passive long-range RFID technology [5]. In particular, we consider tags according to the EPC Class 3 standard. Due to passive RFID technology, sensors draw their energy for communication from the electromagnetical field of the reader, which reduces the power consumption to a minimum and extends their lifetime significantly compared to full-fledged sensor network nodes. Without this complex radio interface and without powerful processing capability these sensors can be much cheaper. Even simple battery-less sensors are conceivable. Without the need to form a complex multi-hop sensor network, these sensors need to be placed only at points of interest. The transmission of readings to infrastructure-based servers uses mobile devices with an integrated RFID reader, as mobile "relays", and the readily available mobile communication infrastructure. With the ongoing development to integrate RFID readers into

Published in Proceedings of the Tenth International Conference on Mobile Data Management: Systems, Services and Middleware (MDM'09), pp. 41-51. Taipei, Taiwan, May 2009.

http://doi.ieeecomputersociety.org/10.1109/MDM.2009.15

mobile devices [6], these devices become a universal tool for inexpensive and autonomous collection of arbitrary sensor data wherever people move. The sensors are provided either by inhabitants in private areas or by the city administration in public areas.

To the best of our knowledge, this work is the first to propose a system that combines cheap RFID-based sensors and mobile devices with an integrated RFID reader for autonomous urban sensing. This novel approach rises a number of interesting challenges that we tackle in this paper: significant energy consumption of read operations and communication of readings, and quality of sensor values in terms of freshness. We address these challenges using three different optimizations. First, we coordinate read operations such that the given update interval is met, but sensors are not read again when they have been recently read. We propose two approaches to coordinate reading: a proactive approach where nodes form an ad-hoc network to manage cooperatively recent update times of sensors, and a reactive approach where the infrastructure schedules read operations. Second, we avoid reading while no sensor is in read range of a mobile node by making nodes aware of the the positions of sensors. Third, we propose an algorithm to avoid collisions due to concurrent reading. At the same time this algorithm optimizes the probability of reading a sensor successfully. We show that our algorithms are effective and perform almost 100% of the possible updates, and they are efficient and require only as little as 10 J/h per node.

The rest of this paper is structured as follows. In Section 2 we present our system model before we provide our algorithms in Section 3. Moreover, in Section 4, we propose an adaptation strategy for adapting the algorithms to different reader densities. In the detailed evaluation in Section 5 we show the feasibility of this approach.

2. System Model

Our system consists of three individual components: *RFID-based sensors*, *reader network*, and *context server*. Readers forward readings of the sensors to the network of context servers where these readings are stored. Next, we describe these components and the underlying assumptions in detail.

2.1. RFID-based Sensors

We assume RFID-based sensors R_i that provide dynamic sensor values. The energy for sensing operations [4] is either provided by the reader through the electromagnetic field (passive RFID tags) during read operations or by an extra battery (semi-active RFID tags). We assume the sensors to be placed statically along the roads of the service area within read range of passing by mobile readers. In addition, we assume that the context server stores the positions of the corresponding sensors.

2.2. Reader Network

The reader network consists of mobile nodes N_i that move randomly along the roads in the service area. Nodes have an integrated RFID reader, a GPS device and a wireless interface for inter-node communication, e.g., an 802.11 interface. The transmission range of the wireless interface is denoted by r_{tx}. In addition, we assume that nodes can communicate with the context servers in the infrastructure using, for instance, GPRS. We assume the clock drift of the mobile nodes during a period of several minutes to be insignificant.

We assume the positions of GPS to be inaccurate. A circle with radius $r_{accuracy}$, which denotes the maximum deviation from the actual position, determines a position A_{pos}. We assume a uniform distribution of the position within A_{pos}.

The communication between sensors and nodes is based on passive RFID technology [5]. The reader powers a sensor so that it can transmit its ID and current value. This reduces the required energy of a sensor to a minimum, however, also the read range r_{read} is reduced to a maximum of about 5 m assuming UHF RFID technology. A reader can read a sensor successfully if it is within the area A_{read} and there is no reader collision [7], i.e, concurrently reading nodes. The area A_{read} is a circle around the sensor with the radius of the read range r_{read}. We assume the read range r_{read} to be of the same order of magnitude as the position accuracy $r_{accuracy}$, but much smaller than the transmission range r_{tx} of a node.

2.3. Context Server

A context server is an infrastructure-based node associated to a service area. It is responsible for managing the readings from the RFID-based sensors in the respective service area. After a mobile node has read a sensor, the mobile node sends the value to the responsible context server. The context server specifies for each sensor an individual update freshness time δ_{cs} of the order of several minutes, which is derived from application requirements. This time defines the requested update interval of the respective sensor. In addition to the δ_{cs} values, the context server manages the position and the time of the last update for each sensor.

3. Algorithms

The main goal of our approach is twofold: First, we want to provide applications with fresh sensor values. Stated formally, sensor value V_R measured by the RFID-based sensor R shall be updated at the context server every δ_{cs} seconds. It should be clear that we cannot give hard guarantees for the freshness because it depends on the distribution of mobile nodes. There might be situations where no node passes by a sensor while the context server has no fresh update.

```
while true do
  doPositionFix()
  if closeToTag() ∧ needUpdate() then
    CROA()
  else
    sleep(DROA())
  end if
end while
```

Figure 1. Main operation of algorithm

Therefore, we aim for a best effort service where a node updates V_R when the context server has no fresh update and it is in the read range of R.

Second, we want to achieve the first goal with as little effort as possible in terms of energy. In particular, we want to avoid unnecessary read operations and communication operations, which are not necessary to provide fresh sensor values. Formally, if RFID sensor R has been read at time t, then we want to avoid further read operations in the interval $(t, t+\delta_{cs})$. Moreover, to reduce the load on the context server and on the mobile devices, we aim to reduce the number of updates, while the context server has a fresh value.

The straight-forward approach to read a sensor reactively whenever it is queried is not effective since it introduces a potentially high delay as no node might be in read range at the query time. In addition, it is not suitable for event-based interaction paradigms where applications want to get a notification whenever a value changes in a certain way, e.g., when the temperature exceeds a certain value. Therefore, our algorithms follow a proactive approach to keep sensor values fresh at all times according to the freshness criteria defined above. Since this approach decouples query processing from update processing, we only consider update algorithms in the following.

Unnecessary read operations are the cause for high energy consumption of mobile nodes. Such operations can occur in three situations. First, a node tries to read a sensor without being in read range. Second, a sensor is read before an update is required w.r.t. the requested update interval δ_{cs}. Third, concurrent readings lead to collisions. Our solutions to avoid these three problems make mobile nodes aware of RFID sensors in their proximity, their respective update times, and concurrently reading nodes.

Our algorithm consists of two concurrent operations. The first operation, early read operation avoidance (EROA) runs concurrently to the second and provides the update time of sensors. Figure 1 shows the second operation. When a node is near to a sensor, the algorithm for concurrent read operation avoidance (CROA) is executed. Otherwise, the algorithm for distant read operation avoidance (DROA) is executed. In the following sections we address these algorithms in detail.

3.1. Early Read Operation Avoidance (EROA)

The idea of Early Read Operation Avoidance (EROA) is to make the mobile nodes aware of the next time a sensor value has to be updated to avoid readings while the server still has a fresh value. To calculate the next update time of a sensor, the node needs to know the last update time and the requested update interval δ_{cs}. The latter is announced to a mobile node by the context server when it enters its service area.

There are two basic approaches to achieve awareness of the last update time of a sensor. The first – proactive management – is based on the idea that nodes cooperatively form an ad-hoc network to manage the update times of sensors. In contrast, the second – reactive management – is based on the idea that the context server in the infrastructure notifies nodes if an update is needed.

3.1.1. Proactive Early Read Operation Avoidance (EROA/P). The basic principle of this algorithm is to store the update time of a sensor at least at some nodes in transmission range r_{tx} of the respective sensor so that nodes in proximity of the sensor can query the update time when needed.

At first we explain our proactive algorithm for managing update times for the case of a single RFID-based sensor R. Afterwards we describe the details of the general case with an arbitrary number of sensors. Our mechanism is based on one hop broadcast messages and on a locally managed list of sensor entries at each node. An entry consists of an ID, position pos, requested interval δ_{cs}, and update time t_{update}. While the ID is used for mapping the entry to a certain sensor, t_{update} is the time of its most recent update.

When a node enters the service area it announces its presence to the context server, which replies with a list of sensors in the service area. From this list, the node initializes a local list with the static values for ID, pos, and δ_{cs}.

When a node successfully reads R it sends an update that includes the sensor value to the context server. In addition, it also signals the t_{update} to its one-hop neighbors via an *Info* message and updates t_{update} locally. As the read range r_{read} is small compared to the transmission range r_{tx}, the node that sends the update is close to the center of the disc around R with radius r_{tx}. Thus, the *Info* message reaches further nodes outside the read range of the sensor. All neighbors that receive this *Info* message locally refresh t_{update}. This mechanism, with minimal cost of one 1-hop ad-hoc broadcast message, prevents nodes within r_{tx} of the sensor from performing an update as long as the server has a fresh update. Therefore, nodes send an *Info* message along with every update.

However, due to mobility, two problems may arise. First, nodes that did not receive the initial *Info* message because the distance between sender and node was greater than r_{tx}

will possibly enter A_{read} of the sensor. At the earliest, this happens after the time $\delta_{cover} = r_{tx}/v_{max}$, where δ_{cover} is the time a node needs to cover a distance of r_{tx} at maximum speed v_{max}. If a node comes into read range before the next update is due, i.e., $\delta_{cover} < \delta_{cs}$, then it should not read. Second, a node that received the initial *Info* message may move out of transmission range and thereby miss a duplicate update. Thus, a node needs to check the validity of t_{update}. Our refresh mechanism handles these two problems.

The basic idea of this refresh mechanism is to store and keep the update time at nodes within transmission range r_{tx} of the sensor rather than only broadcasting t_{update} once when the sensor is read. Nodes in read range r_{read} of the sensor then can query the update time with 1-hop broadcast messages. To assess the validity of the local t_{update} and to decide whether it needs to be refreshed, nodes manage the time t_{com} of the last communication related to a sensor.

The anticipated update time t_{aut} specifies the earliest point in time for an update of the sensor, as locally seen by a node. We define t_{aut} as follows:

$$t_{aut} = t_{update} + \delta_{cs} \tag{1}$$

A node triggers the refresh mechanism before it starts to read, i.e, when it is close to the sensor (see Section 3.3). A node verifies if t_{aut} is a future value. In addition, the node checks the length of time δ_{idle} since t_{com}. If $\delta_{idle} < \delta_{cover}$, the node assumes that its update time is valid because it was in the transmission range around the sensor for a period of δ_{cover}. Otherwise, the node may have missed a duplicate update while outside transmission range. Therefore, it queries the 1-hop neighbors for t_{update} by sending a *Query* message.

In the *Query* message, a node specifies the ID and the respective local t_{update}. All nodes that receive a *Query* compare their local entry to the received update time. If a node has a more recent update time it sets a timer to send an *Info* message as reply. The reason to postpone the reply is twofold. First, a random jitter is need to reduce collisions through simultaneous replies. Second, the *Info* with the most relevant information, i.e., most recent time, should be sent. Therefore, a node chooses a small random jitter between $[0, j]$ if it knows that no update is needed. The value of j is chosen according to the read interval (see Section 3.3). Otherwise, it chooses a larger random jitter between $]j, 2j]$. On receiving an *Info*, a node cancels its own timer, if it cannot contribute a more recent update time. With this mechanism few messages are sent per refresh cycle and the number of cycles for a specific update interval is limited by $\delta_{cs}/\delta_{cover}$.

The cost, i.e., the number of messages, to manage t_{update} with our proactive ad-hoc algorithm increases with node density and with the update interval. In Section 3.1.2 we present our reactive algorithm for managing t_{update}, which especially suits large δ_{cs} values, whereas we address the problem of high node density in Section 4.

For the case of multiple sensors we now present the details of the generalized proactive algorithm. Although it is effective to apply the algorithm described above to each individual sensor, it is more efficient to bundle multiple update times into a single *Info* message to reduce the message overhead and outdated information. Instead of replying immediately, a node that receives a *Query* for a sensor sets a timer and adds its reply to a list. If the node then receives another *Query* while the timer is already set, it adds the reply to the list of replies. On receiving an *Info* message with a more recent t_{update} it removes its own reply from the list. It only cancels its timer if its list is empty. When the timer expires, it sends all valid update times with one single *Info* message. Since an update time is only valid if the equation $\delta_{idle} < \delta_{cover}$ holds, the size of the reply message is limited by the number of sensors on the disc centered at the node with radius r_{tx}.

3.1.2. Reactive Early Read Operation Avoidance (EROA/R). In contrast to the proactive EROA/P algorithm, this algorithm relies on the management of update times at the context server in the infrastructure.

The context server notifies nodes within a maximum radius of the sensor to perform an update when needed. It periodically repeats this notification until an update is performed. With this mechanism nodes ignore sensors as long as they recently have been notified to read it. In addition, this algorithm guarantees that no node reads when the context server has a fresh update. In contrast, EROA/P may cause early read operations if the time of the last update is lost.

A notification message *Notify* includes the ID, the position pos, and the notification radius r_{notify} for the corresponding sensor. A node ignores the *Notify* message if its distance to the sensor is larger than r_{notify}, otherwise the sensor is marked as active and will be read when the node enters its read range. We set $r_{notify} = r_{tx}$, i.e., the transmission range. In addition, we set the notification interval δ_{notify}, i.e., the time between two successive notifications, to the value of δ_{cover}. This ensures that a new notification is only sent when nodes possibly come into read range that have not been notified already.

A node resets the state of a sensor to inactive, when the latest *Notify* was received more than δ_{notify} ago or, when it detects that its distance to the sensor exceeds r_{notify}. In both cases a node assumes that it will be notified again when it enters the notification area while the sensor still needs to be updated. Moreover, nodes can ignore the sensor as soon as they receive an *Info* message indicating an update.

The cost of this reactive algorithm for management of update times is independent of the required update interval δ_{cs}. In contrast, the cost of the proactive algorithm presented in Section 3.1.1 grows with the required update interval. For details about the energy consumption of both algorithms we

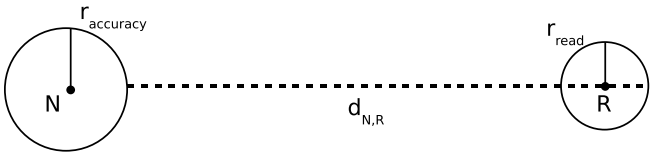


Figure 2. Distance metric $d_{N,R}$

refer to Section 5 where we present the evaluation results.

3.2. Distant Read Operation Avoidance (DROA)

Next, we tackle the problem of distant read operations that occur whenever a node tries to read while being outside the read range. The basic idea is that nodes determine their position using a GPS device. By comparing this position to the known positions of sensors, which are sent to a node by the context server, a node can determine its proximity to sensors. However, a positioning technology like GPS has two challenging characteristics (see Section 2 for details): high energy consumption and inaccurate position. Our algorithm reduces the number of position fixes by computing the time a node can deactivate positioning before it may pass by a sensor. This calculation has to be done carefully such that nodes may not pass by a sensor unnoticed. In particular, we have to consider the inaccuracy of node positions.

At first we define the minimum distance a node N has to cover before passing by the read range of a sensor R as

$$d_{N,R} = |pos_{GPS}(N) - pos(R)| + r_{read} - r_{accuracy} \quad (2)$$

$pos_{GPS}(N)$ denotes the GPS position of N; $pos(R)$ is the real position of R. A node adds r_{read} to the distance to R, because reading is possible as long as the node is in read range after passing by a sensor. Moreover, due to the inaccuracy of GPS, N might actually be $r_{accuracy}$ closer to R (see Figure 2).

This definition allows for exceptional cases where nodes are in read range of a sensor but do not notice. This can happen when nodes turn while they are in read range of a sensor and when nodes walk past the sensor. While both cases are only relevant for nodes that almost move at v_{max}, the latter is, in addition, unlikely since we assume sensors to be placed directly where nodes pass by. We show the insignificance of these effects in the evaluation (see Section 5).

$$\delta_1 = d_{N,R}/v_{max} \quad (3)$$

δ_1 is the time a node needs to pass the distance of $d_{N,R}$ at maximum speed. Node N can turn off positioning after a position fix at t_{fix} until $t_{fix} + \delta_1$. If the next reading is scheduled at $t_{aut} > t_{fix} + \delta_1$, N can postpone the next fix even longer. How t_{aut} is derived is explained in Section 3.1.

$$\delta_2 = max(t_{aut}(R) - t_{now}, 0) \quad (4)$$

$\delta_{min} \leftarrow \infty$
for all RFID_Sensor R **do**
  $\delta_1 \leftarrow d_{N,R}/v_{max}$
  $\delta_2 \leftarrow max(t_{aut}(R) - t_{now}, 0)$
  $\delta_{fix} \leftarrow max(\delta_1, \delta_2)$
  if $\delta_{fix} < \delta_{min}$ **then**
    $\delta_{min} \leftarrow \delta_{fix}$
  end if
end for
return δ_{min}

Figure 3. DROA: Computation of position fix interval

$$\delta_{fix} = max(\delta_1, \delta_2) \quad (5)$$

δ_2 is the time until the next update is needed. The larger of these values δ_{fix} defines the time span to turn off positioning for the respective sensor. Note that $t_{aut}(R)$ might change when another node reads R. In this case, a node updates t_{aut} accordingly without performing a position fix.

For multiple sensors, a node can only switch off positioning for the minimum δ_{fix} (see Figure 3).

When δ_{cs} is large, δ_{min} is mainly determined by δ_2. Otherwise, it is mainly determined by δ_1. Moreover, the most restrictive sensor determines the position fix interval. These considerations influence the CROA algorithm for efficient reading as presented in Section 3.3.

Due to position inaccuracy, a node cannot definitely determine whether it is in read area A_{read}. We define the *target area* A_{target} of a sensor as the area where the probability of successfully reading is larger than zero. This area is a disc centered at R with radius $r_{target} = r_{read} + r_{accuracy}$. When a node detects to be in A_{target} and an update is needed it switches to the CROA algorithm (see Section 3.3). It switches to the DROA algorithm when it is outside A_{target}.

3.3. Concurrent Read Operation Avoidance (CROA)

When multiple nodes read concurrently, a read collision occurs that results in unsuccessful readings. However, high reader density can be exploited to increase the probability of successful reading. The basic idea of our approach is to suppress reading, based on the distributed computation of the probability of successful reading by letting nodes signal their read operations to their 1-hop neighbors in the ad-hoc network.

After each position fix, a node determines the set of sensors that need to be updated and whose target areas A_{target} cover its position. Then, it checks if any of these sensors R_j has to participate in reading. Therefore, a node computes its individual probability $p_{success}$ of reading sensor R_j successfully and the probability p_{group} of R_j being read successfully by the group of currently reading nodes. A node reads if $p_{success} > p_{min}$ and $p_{group} < p_{max}$ is fulfilled

for at least one sensor. A node cyclically performs these tasks with an interval δ_{read}. This interval depends on the maximum node speed and allows to trade-off effectiveness and efficiency of reading. When the interval is large, nodes risk passing through A_{read} without reading; when the interval is large, nodes read twice at almost the same position.

$p_{success}$ is the probability of a node being within r_{read} of a sensor, i.e, the overlapping zone as indicated in Figure 4. In Figure 4a, node N has a fairly low probability of successfully reading R. Whereas in Figure 4b the probability is maximal. Note that the maximum may be below 1 according to the accuracy.

The equation to compute the probability $p_{success}$ for a specific inaccurate position A_{pos} and the reading area of a sensor A_{read} is the following:

$$p_{success} = \frac{A_{pos} \cap A_{read}}{A_{pos}} \tag{6}$$

Although we only consider uniformly distributed positions, this idea is applicable to more sophisticated models as well.

To compute the success probability p_{group}, a node sends, directly before each read operation, a *Beacon* message as 1-hop broadcast to its neighbors. This message includes the position of the node and, implicitly, the time when the node reads. By sharing this information, every receiver can compute $p_{success}$ for the respective node. Moreover, a node can schedule its own reading so that no collisions occur. A node computes $p_{success}$ for all nodes of which it is aware, and that are closer to the sensor than itself. A node is only aware of other nodes that recently signaled their reading with a *Beacon* message. According to the following equation a node then computes the probability p_{fail} that none of these k nodes succeeds in reading:

$$p_{fail} = \prod_{i=1..k} 1 - p_{success}(i) \tag{7}$$

When the probability $p_{group} = 1 - p_{fail}$ is lower than p_{max} the node sends a *Beacon* message and tries to read the sensor. The full algorithm is listed in Figure 5. A node switches to the DROA algorithm when an update is performed or when it leaves A_{target}.

The effect of this algorithm is that the number of concurrently reading nodes is limited and nodes closer to a sensor, i.e., with higher success probability, are preferred for reading.

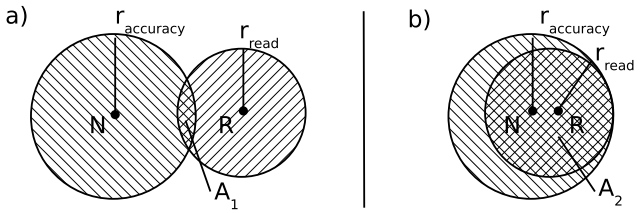


Figure 4. Probability of successful reading

```
while doPositionFix() ∈ A_target ∧ needUpdate() do
  p_success ← successProbability(INDIVIDUAL)
  p_group ← successProbability(GROUP)
  if p_success > p_min ∧ p_group < p_max then
    send(BEACON)
    if read() = SUCCESS then
      sendUpdate()
    end if
  end if
  sleep(δ_read)
end while
```

Figure 5. CROA: Concurrent read operation avoidance

4. Adaptive Early Read Operation Avoidance (AEROA/P)

The idea of the AEROA/P is to reduce the number of participating nodes in the reading process to the minimum needed to fulfill effectively the requested update interval δ_{cs}.

Node movement, i.e., the number of nodes passing by a sensor during a certain period of time f_{pass}, defines the upper bound for the minimum possible update interval $\delta_{cs}^{'}$. For $\delta_{cs} > \delta_{cs}^{'}$ a smaller number of passing nodes would be sufficient. Since we cannot constrain the physical node movement, we artificially reduce f_{pass} by assigning only a subset of all sensors to certain nodes.

EROA/P is effective and efficient for an arbitrary number of nodes passing a sensor. When f_{pass} is high, a sensor is updated when needed. This behavior allows to trade off between timeliness of updates and energy savings. We adapt EROA/P by monitoring the history of the latest update times at the context server and by determining the average update interval $\delta_{average}$ as indicator for the number of passing nodes.

We reduce the effective value of nodes passing a sensor and, thus, the overall energy consumption, by allowing nodes to ignore the respective sensor in the DROA algorithm (see Section 3.2). Since this algorithm computes the time to deactivate positioning based on the most restrictive sensor, ignoring sensors allows for longer periods of deactivated positioning and, therefore, energy savings.

The initial assignment of sensors to a node (see Section 3.1 for details) is the mechanism we use to adapt the sensor density of a node and thereby the value of f_{pass}. This mechanism can be used to adapt sustainably but inertly the number of nodes passing by.

When nodes enter the service area, they request the list of sensors from the context server (see Section 3.1.1). We now add to each entry of this list an *ignore* flag that indicates if the node may ignore the respective sensor in the DROA algorithm. We do not remove the sensor from this list

to allow nodes to further participate in the EROA/P and CROA algorithms, because energy consumption for these algorithms is fairly low compared to positioning in sparse sensor environments. This mechanism allows for assigning different sets of sensors to different nodes. Therefore, each sensor is assigned an *ignore* flag at the context server. This flag specifies whether the corresponding sensor is assigned to nodes. We set *ignore* as follows:

$$ignore = \begin{cases} true & \text{if } \delta_{average} < \delta_{cs} + delay_{th}, \\ false & \text{else.} \end{cases} \quad (8)$$

The number of nodes in the service area changes over time, since nodes continuously enter and leave this area. This mechanism only affects those entering. Therefore, to adapt rapidly, all nodes that enter the service area ignore a sensor, if the *ignore* flag is set. This is necessary because δ_{cs} is fairly small compared to the average time a node stays in the service area. The value of $delay_{th}$ specifies the maximum accepted delay of updates and, thus, allows for saving energy through extended intervals of deactivated positioning.

5. Experimental Evaluation

In this section we present our simulation model followed by the results of extensive simulations of our algorithms. We implemented our algorithms for the network simulator ns-2. In the following we refer to the following implementations:

- **EROA/P:** This implementation manages update times proactively with EROA/P (see Section 3.1.1), detects proximity to sensors with DROA (see Section 3.2) and reads sensors efficient with CROA (see Section 3.3).
- **EROA/R:** In contrast to the EROA/P implementation, the EROA/R implementation manages update times reactively (see Section 3.1.2) instead of proactively.
- **AEROA/P:** In addition to the EROA/P, the AEROA/P performs adaptation as described in Section 4. The $delay_{th}$ is set to 20% of δ_{cs}. The average time a node stays in the service area is set to 15 min. The numbers of nodes leaving and entering the service area are equal.
- **Isolated:** A simple isolated approach where nodes are unaware of the required update interval and read when they are close to a sensor. Moreover, nodes implement the DROA algorithm (see Section 3.2). After an update a node skips reading for ten seconds. This implementation presents the worst case for duplicates.
- **Global:** The Global approach is implemented to compare our approach with the best case where nodes access global knowledge to perform only the necessary updates.

We implemented our algorithms using the 802.11b extension of ns-2 with the transmission range r_{tx} set to 100 m. The size of the service area is set to 1000 m x 1000 m, which is sufficiently large considering the locality of the evaluated algorithms. More important is the effect of node density which we evaluate in a wide range. The nodes move in the service area according to a graph based mobility model [8] on the road graph of the city of Stuttgart. Nodes choose a random speed. By default, the maximum node speed is 3 m/s. 25 sensors are randomly distributed on the service area. The RFID read range r_{read} is set to 5 m [5]. According to [9] we set the duration of reading, i.e., the time to transmit a sensor reading, to δ_{read} to 20 ms. The default position accuracy is set to 5 m. Each simulation is performed 10 times and lasts 3600 seconds.

Table 1. Energy Model

Component	Energy [mJ]
GPS [10]	
Position Fix	75
RFID [9]	
Read	80
802.11b at 1 Mbps [11]	
(broadcast rate)	
Send (1000 Bit)	2
Receive (1000 Bit)	1
GPRS [12]	
Send (1000 Bit)	80
Receive (1000 Bit)	40

To measure the energy consumption of the battery powered mobile nodes we rely on the energy model given in Table 1, which also provides references for the different values.

5.1. Percentage of Duplicates

In this section we evaluate the efficiency of the algorithms in terms of duplicate updates. An update is valid for the time δ_{cs}. We measure the time δ_{valid} for that the context server has valid updates during a simulation run. In addition, we measure the number of updates $U_{measure}$ the mobile nodes performed in order to provide the context server with valid updates for this time span. We compute the minimum number U_{min} of updates to provide valid updates for the same timespan as follows:

$$U_{min} = \delta_{valid}/\delta_{cs} \quad (9)$$

Based on U_{min} we define the percentage of duplicates (POD) as follows:

$$POD = 1 - \frac{U_{min}}{U_{measure}} \quad (10)$$

At first, we plot the percentage of duplicates POD in Figure 6 for a different numbers of nodes in the network. In this scenario δ_{cs} is set to 4 minutes. EROA/R produces, independent of the node density, practically no duplicates. This is caused by the reactive mechanism, which prevents

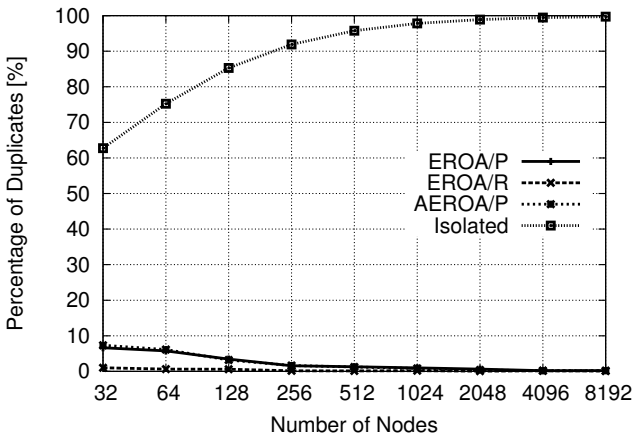


Figure 6. POD for different numbers of nodes

nodes from reading as long as they receive a notification. It also shows the effectiveness of the invalidation of notifications when an update is performed. EROA/P and AEROA/P behave similarly. However, effectiveness in suppressing duplicates depends on the node density. With lower node densities, the cooperative management of update times is less effective. Thus, nodes sometimes do not know about previous updates and have to perform redundant read operations. However, the percentage of duplicates does not exceed 7% for EROA/P and AEROA/P even with, on average, only one node within the transmission range r_{tx} of each sensor (32 nodes: $32 * Pi * (r_{tx})^2/A = 1$). The Isolated approach fails to prevent duplicate updates, since nodes are unaware of previous updates. Even for a relatively small number of 32 nodes the number of duplicates reaches 60% since still multiple nodes pass by the same sensors shortly after each other.

Figure 7a depicts the percentage of duplicates for varying values of the requested update interval t_{cs}. The number of nodes in the network is 256. EROA/R produces, independent of the requested update interval δ_{cs}, practically no duplicates. The POD for EROA/R and AEROA/P increases slowly with growing values of δ_{cs}, since the cooperative management of update times is less reliable for longer update intervals. Still, Figure 7a indicates that the effectiveness of our lightweight cooperative approach degrades gracefully. The absolute number of updates for the Isolated approach is independent of δ_{cs} at a high level and, thus, the percentage of duplicates grows.

In Figure 7b we present the POD values for different position accuracies. In this case, EROA/R, EROA/P, and AEROA/P behave similarly. The increasing POD with high inaccuracy is caused by the fact that the size of target regions grows and, therefore, these regions can overlap. While trying to read a certain sensor, a node may unintentionally read another sensor, possibly resulting in a duplicate update. The drop of the POD for the Isolated approach is due to the increased number of collisions, which prevents nodes from successfully reading and updating a sensor. For different sensor densities, the same effect can be observed. However, the overlapping of target regions is not due to growth in size but because of growth in the number of the target regions. In both cases, these duplicates do not put additional load on the nodes, because they are a byproduct of the requested reading.

Figure 7c plots the POD for different values of the maximum node speed v_{max}. The POD for the EROA/P and AEROA/P approach increases with growing speed of the nodes. This is caused by the lower average time nodes stay in transmission range of a sensor and, therefore, the probability of losing the most current update time.

The EROA/R algorithm causes almost no duplicates at all. The only duplicate updates are performed when a node tries to read a specific sensor and unintentionally succeeds in reading a different. However, no readings are wasted. In addition, EROA/P produces duplicates, as the evaluation shows, due to the loss of the most recent update time. This effect was expected for large update intervals δ_{cs} as well as for increased node speed. However, only a small number of duplicates are performed for a wide range of node densities. This is due to the self-tuning of EROA/P. In the case of low node density, the POD is small because nodes only seldomly pass by a sensor. With high node density, the POD is even lower, because EROA/P profits from an increased redundancy of nodes that manage t_{update}.

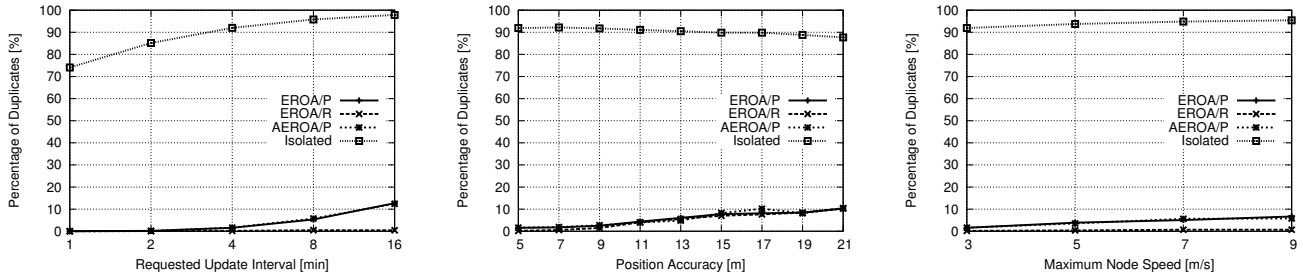
5.2. Update Validity

The previous evaluations showed that our update protocols reduce the number of redundant updates significantly. Now, we show that despite the reduced number of duplicates, we still achieve the desired update frequency. We measure the time δ_{valid} for which the context server has fresh, i.e, valid, updates during a simulation run δ_{sim}.

$$UV = \frac{\delta_{valid}}{\delta_{sim}} \tag{11}$$

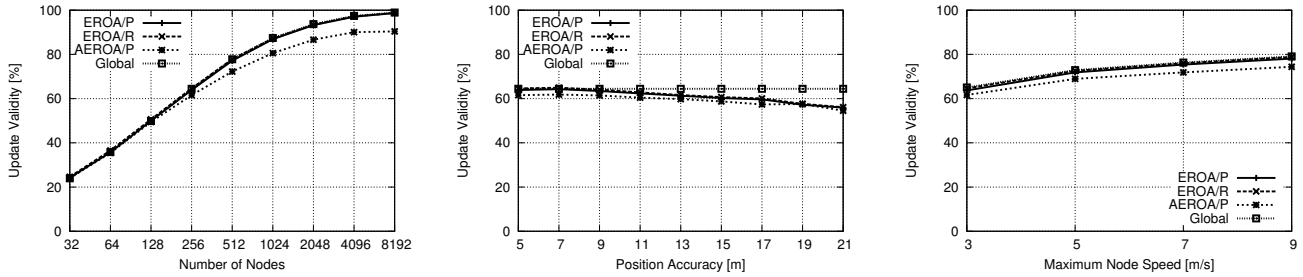
The UV metric allows to compare the effectiveness of our approaches with the Global approach, which exploits global knowledge, and, therefore, does not miss any updates. By comparison we can determine the number of missed updates of our approaches.

Figure 8a depicts the UV for experiments with varying number of nodes. The δ_{cs} is reduced to 1 min to evaluate the effectiveness in case of stressing conditions. Since the EROA/P and EROA/R approaches perform as well as the Global approach, we can infer that no updates are missed. The AEROA/P approach performs almost equally well up to the scenario with 256 nodes. Here, the adaptation mechanism starts to reduce the number of nodes assigned to specific



a. Update interval δ_{cs} b. Position accuracy $r_{accuracy}$ c. Maximum speed v_{max}

Figure 7. Percentage of Duplicates POD



a. Number of nodes b. Position accuracy $r_{accuracy}$ c. Maximum speed v_{max}

Figure 8. Update Validity UV

sensors. Therefore, updates may be delayed, as specified by $delay_{th}$, and the UV value slightly degrades.

To evaluate the relation between UV and the position accuracy we tested with $r_{accuracy}$ ranging from 5 m to 21 m. Figure 8b shows that AEROA/P performs slightly worse than the other approaches due to the $delay_{th}$ as explained above. Moreover, Figure 8b shows a slight decrease of the UV, which indicates that few updates are missed. Due to the large inaccuracy, the success probability $p_{success}$ of a node to read a sensor is limited. However, nodes may skip reading if $p_{success}$ is below a threshold, a trade-off between energy savings and actual read success probability.

With growing node speed, the frequency of nodes passing by a sensor is increased. This leads to an increase of the UV as depicted in Figure 8c. EROA/P and EROA/R do not miss updates and behave like the Global approach. AEROA/P performs slightly worse as specified by $delay_{th}$.

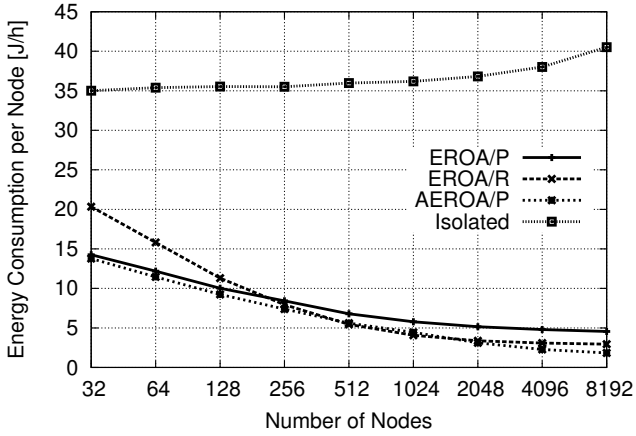


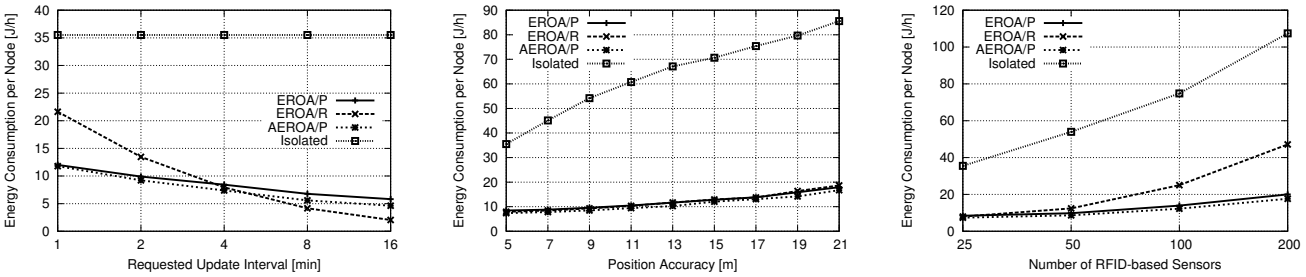
Figure 9. EC for different numbers of nodes

5.3. Energy Consumption

Our optimizations reduce the energy spent by mobile nodes by reducing the number of read operations, communication and for position fixes. Next, we quantitatively evaluate the energy savings achieved.

The average energy consumption (EC) is the energy a node consumes in an hour; it is measured in Joules. It is the sum of the energy consumed by GPS, RFID reader, and for sending and receiving for both 802.11b and GPRS.

In Figure 9 the EC is depicted for scenarios with different numbers of nodes. The Isolated approach shows the expected constant energy consumption for small numbers of nodes. However, the higher the number of nodes, the higher is the EC, due to the increasing number of collisions. The EROA/P approach profits from growing numbers of nodes, because the load is distributed over a larger set of nodes. The EC for 8192 nodes is only about 33% the EC for 32 nodes. The adaptive AEROA/P approach benefits even more from high node densities. It only consumes about 15% of the energy in case of 8192 nodes compared to 32 nodes. This is because of the adaptive reduction of nodes that participate



a. Requested update interval δ_{cs} b. Position accuracy $r_{accuracy}$ c. Number of sensors

Figure 10. Energy Consumption EC

in updating a sensor. The EROA/R approach benefits from higher node density, because the time between notification and update is reduced. After an update, the node can ignore the sensor again. Figure 10a depicts the energy consumption EC with growing requested update interval δ_{cs}. The EC for the reactive and the proactive approaches are roughly equal. For lower values of δ_{cs}, the EROA/P and AEROA/P approaches have lower EC than EROA/R because of the high energy consumption of GPRS, which is used to notify nodes. The EC of EROA/R scales with the number of updates, while the EC of the EROA/P depends on the length of δ_{cs} (see Section 3.1).

The energy consumption EC of a node also depends on the position accuracy $r_{accuracy}$. Figure 10b shows that the average EC for our approaches is increased by 100% when $r_{accuracy}$ is incremented from 5 m to 21 m. This is due to the increased number of unsuccessful readings. The increase is higher for the Isolated approach because it performs more updates and, thus, the impact of RFID readings on the EC is higher compared to our approaches.

The overall number of updates grows with the number of sensors in the service area. As the EC of the EROA/R approach scales with the number of updates it shows an increase with growing number of sensors as depicted in Figure 10c. For the EROA/P and AEROA/P approaches the EC increases much slower.

The EC of the EROA/R algorithm is independent of δ_{cs} when considering the energy consumption per update. However, it depends on the node density. With high node density, an update can be performed shortly after the notification, which allows nodes to ignore the respective sensor again. The EROA/P shows a similar decrease of the EC. However, the efficiency of EROA/P degrades with growing requested update interval δ_{cs}. The different characteristics between EROA/P and EROA/R explain the point of balance between the EC values of the two approaches that we found in the evaluated scenario to be δ_{cs} = 4 min in case of 256 nodes in the network. AEROA/P showed to reduce the EC up to 50% by allowing for delayed updates.

The most significant energy consuming task of our algorithms is positioning. Mechanisms such as dead reckoning or map matching could reduce the EC. However, these optimizations work independently of our algorithms and, therefore, are outside the scope of this paper.

5.4. Summary

One major goal of this paper is to provide algorithms that perform all possible requested updates. The evaluation results indicate this goal is achieved. Only for increased inaccuracy of position information, are some updates missed. This is caused by the CROA algorithm, which balances efficiency and effectiveness. Another goal of this paper is efficient updating. As the evaluation shows, this goal is achieved too. The number of duplicates is low and the energy consumption of the nodes is far beyond the Isolated approach. Moreover, a typical mobile phone battery allows for hundreds of hours urban sensing.

6. Related Work

Environmental sensing is currently a very active topic in different research fields. The research field of sensor networks mainly focuses on the autonomous monitoring of environmental parameters in inaccessible areas. However, most similar to our scenario are projects such as [2], [13] that use sensor nodes for environmental sensing. Due to the high price of sensor nodes and the high density requirements, these approaches also imply high cost for large-scale deployments. In addition, these approaches suffer from the battery depletion problem.

Another set of approaches is based on instrumented mobile devices. These approaches exploit the resources of mobile devices. Gellersen et al. [14] propose the integration of sensors into mobile devices to achieve direct context awareness of mobile devices rather than to use mobile devices for context data collection as [15], [3]. Rudman et al. [15] attach sensors for monitoring air pollution to a tablet PC. MobGeoSen [3] is based on the integration of sensors to mobile phones, which are carried by a large number of people. Although we agree on the advantages of mobile phones for environmental sensing, the integration of sensors

has several disadvantages. Due to size, cost and technical reasons, only a limited number of sensors can be integrated. Moreover, the need for attaching sensors to a mobile phone is often too cumbersomely and, thus, restricts the number of people participating in environment sensing.

In the field of data-centric storage Ratnasamy et al. [16] propose a geographic hash table (GHT) for dissemination of data to specific locations. Although we also aim for maintaining data at specific locations, GHT is proposed for static environments and, therefore, not applicable to MANETs. Zahn et al. [17] propose a distributed hash table (DHT) for MANETs. However, the overhead for maintaining a DHT in a dynamic environment is too high for the infrequent data access rates in our scenario. In general, DHTs do not consider geographic proximity. Early work of us [18] deals with location based storage and migration mechanisms for maintaining data at specific locations in MANETs. Although [18] allows for storing data close to specific locations, it requires frequent position fixes for geographic routing.

The algorithmic aspects of our paper are also loosely related to multi-reader coordination in RFID-based systems [19], [7]. However, these approaches tackle the goal to increase read throughput instead of optimizing energy efficiency of read operations. Moreover, they assume a completely different system model with static readers instead of mobile readers.

7. Conclusions

In this paper, we presented a novel scenario for environmental sensing based on the combination of simple and cheap RFID-based sensors and mobile devices such as mobile phones with integrated long range RFID readers. In our system, the mobile nodes cooperatively read sensors installed in the environment as they pass by and transmit the data to a infrastructure of context servers. We have presented algorithms to achieve quality requirements in terms of update frequency and efficiency in terms of energy consumption. In our system, mobile nodes form an ad-hoc network for the cooperative management of update times and the coordination of reading to avoid redundant readings and collisions. Besides a decentralized algorithm for coordinating read operations, we have shown a complementary algorithm that exploits infrastructure based coordination. By extensive simulations, we have shown the effectiveness as well as the efficiency of our algorithms. With our approaches, typical batteries of mobile nodes allow for hundreds of hours of operation performing nearly 100% of the possible updates.

Acknowledgements

This work is funded by the German Research Foundation within the Collaborative Research Center 627 (Nexus).

References

[1] D. Cuff, M. Hansen, and J. Kang, "Urban sensing: out of the woods," *Commun. ACM*, vol. 51, no. 3, pp. 24–33, 2008.

[2] C. Chen and J. Ma, "Mobile enabled large scale wireless sensor networks," *Advanced Communication Technology. ICACT 2006. The 8th Intl Conf*, vol. 1, pp. 333–338, Feb. 2006.

[3] E. Kanjo, S. Benford, M. Paxton, A. Chamberlain, D. S. Fraser, D. Woodgate, D. Crellin, and A. Woolard, "Mobgeosen: facilitating personal geosensor data collection and visualization using mobile phones," *Personal Ubiquitous Comput.*, vol. 12, no. 8, pp. 599–607, 2008.

[4] H. Deng, M. Varanasi, K. Swigger, O. Garcia, R. Ogan, and E. Kougianos, "Design of sensor-embedded radio frequency identification (se-rfid) systems," in *Proc. of the 2006 IEEE Int. Con. on Mechatronics and Automation*, 25-28 June 2006.

[5] M. Beuttner and D. Wetherall, "An Empirical Study of UHF RFID Performance," in *Proc. of MobiCom 2008*, 2008.

[6] NFC Forum, Sep. 2008. [Online]. Available: www.nfc-forum.org

[7] J. Ho, D. Engels, and S. Sarma, "Hiq: a hierarchical q-learning algorithm to solve the reader collision problem," *Int. Symp. on Applications and the Internet Workshops, 2006.*, pp. 88–91, Jan. 2006.

[8] I. Stepanov, P. J. Marron, and K. Rothermel, "Mobility modeling of outdoor scenarios for MANETs," in *Proc. of ANSS 38, San Diego, USA*, April 2005, pp. 312–322.

[9] Skyetek, Sep. 2008. [Online]. Available: skyetek.com/Portals/0/Documents/Products/SkyeModule_M9_DataSheet.pdf

[10] Navman, Sep. 2008. [Online]. navman.com/Documents/OEM_docs/Jupiter30/LA000576C_Jupiter30_DataSheet.pdf

[11] Summitdatacom, September 2008. [Online]. www.summitdatacom.com/Documents/sdc-cf10g Product Brief 200803.pdf

[12] B. Gedik and L. Liu, "Mobieyes: A distributed location monitoring service using moving location queries," *IEEE Trans. on Mobile Computing*, vol. 5, no. 10, 2006.

[13] J. Beutel, O. Kasten, F. Mattern, K. Rmer, F. Siegemund, and L. Thiele, "Prototyping wireless sensor network applications with btnodes," in *In Proc. 1st European Workshop on Sensor Networks (EWSN 2004)*. Springer, 2004, pp. 323–338.

[14] H. W. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artifacts," *Mob. Netw. Appl.*, vol. 7, no. 5, pp. 341–351, 2002.

[15] P. Rudman, S. North, and M. Chalmers, "Mobile pollution mapping in the city," in *Proc. UK-UbiNet workshop on eScience and ubicomp, Edinburgh*, May 2005.

[16] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with GHT, a geographic hash table," *Mob. Netw. Appl.*, vol. 8, no. 4, pp. 427–442, August 2003.

[17] T. Zahn and J. Schiller, "MADPastry: A DHT Substrate for Practicably Sized MANETs," in *Proc. of 5th Workshop on Applications and Services in Wireless Networks (ASWN2005)*, Paris, France, June 2005.

[18] D. Dudkowski, P. J. Marron, and K. Rothermel, "Migration policies for location-centric data storage in mobile ad-hoc networks." in *Proc. of MSN'07*. Springer, 2007.

[19] N. Vaidya and S. R. Das, "Rfid-based networks: exploiting diversity and redundancy," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 12, no. 1, 2008.