Energy-efficient Tracking of Mobile Objects with Early Distance-based Reporting

Tobias Farrell Ralph Lange Kurt Rothermel
Universität Stuttgart,
Institute of Parallel and Distributed Systems,
Universitaetsstr. 38, 70569 Stuttgart, Germany
<*firstname.lastname*>@ipvs.uni-stuttgart.de

***Abstract*—Many location-based systems rely on fine-grained tracking of mobile objects that determine their own locations with sensing devices like GPS receivers. For these objects, energy is a very valuable and limited resource. A distance-based reporting protocol can be employed to reduce the energy they consume by sending position updates. However, the energy required for position sensing has not been considered in the past.**

In this paper, we study how the resulting energy consumption from both sensing and update operations can be reduced for distance-based reporting. We show that significant savings are achieved by sending position updates earlier than actually required. For uniform movement, we derive the minimal power consumption analytically. Subsequently, two novel online heuristics are proposed that control the sending of position updates at runtime. Their effectiveness is validated by extensive simulations.

I. Introduction

Due to advances in wireless communications and low-cost positioning devices like GPS receivers, location-based systems have attracted tremendous interest lately [4]. A wide range of applications has been identified, including location-aware information services, asset tracking, or fleet management.

Often those applications rely on fine-grained location information from a potentially large number of mobile objects. For that reason, a so-called *location manager (LM)* keeps track of the current positions of the *mobile objects (MOs)* and allows applications to access this information. Each MO is equipped with a positioning sensor to determine its position and communicates this data to the LM over a wireless network.

Substantial research has been conducted to increase the efficiency, robustness and scalability of a (distributed) LM (e.g., [8]). Another area of research addresses the interaction of MOs with a remote LM node. For updating the position information a variety of *reporting protocols* have been proposed [1],[7],[16]. They minimize the number of update messages while guaranteeing a bounded accuracy of the location information maintained by the LM. This offers three advantages: saving radio bandwidth, reducing the load of LM nodes, and reducing the energy consumption of MOs.

For many MOs, such as cellular phones or PDAs, energy is the most precious resource, due to limited battery capacities. Thus, minimizing the energy consumption is of particular importance. However, all reporting protocols described in the literature are based on the assumption that communication is the only relevant factor to consider. Clearly, under this assumption most energy can be conserved by minimizing the number of update messages. But, we argue that this assumption oversimplifies matters, at least for the prominent sensing technology GPS. For example, at a sensing rate of 1 Hz, a GPS receiver consumes 80 mJoules per second [12]. The same amount of energy is required by GPRS to transmit 1 kilo-bit of data [3]. Consequently, the design of an energy-efficient reporting protocol should consider the energy consumed by position sensing, too.

In this paper, we focus on one particular reporting protocol, called *distance-based reporting (DBR)* [7]. With this protocol an MO updates the position stored by the LM whenever its current position deviates from the previous update by more than a given threshold. This threshold depends on the required accuracy of the remote position data. Consequently, this protocol reduces the number of messages by sending updates as late as possible. However, we will show that this is far from optimal in terms of energy consumption, if GPS or a similar sensing technology is used. We will see that energy consumption can be decreased by updating the position earlier, i.e., before the threshold is reached. This is because the frequency of position sensing can be reduced significantly by early updates. But at the same time, this increases the number of updates messages to be sent. Therefore, the interesting question is when to update the position so that the total energy consumption is minimized.

The answer to that question depends strongly on how the MO moves after the decision. For uniform movement we will provide an analytical model for computing the optimal time to send an update. For arbitrary movement we will propose two heuristics, one of them based on movement prediction. Our evaluation shows that they achieve saving more than 50% of the energy consumed by both sensing and update operations.

While our results apply to distance-based reporting, we are convinced that the same principle can be applied to other reporting protocols as well – including dead-reckoning reckoning protocols, which also use a movement prediction [1],[9],[17].

In summary, the contributions of this paper are:

- *Early Distance-based Reporting (EDBR),* a novel reporting protocol to reduce the total energy consumption of position sensing and update operations;
- An analytic model of EDBR for uniform movement;
- Two heuristics for EDBR that decide at runtime when to send the next position update.

Published in Proceedings of the 4th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '07). Philadelphia, PA, USA. August 2007
© IEEE 2007
http://dx.doi.org/10.1109/MOBIQ.2007.4450984

The remainder of this paper is structured as follows: First, the underlying system model is discussed in Sec. II. Then, we present the basic idea of Early Distance-based Reporting in Sec. III. Subsequently, Sec. IV provides an analytic model of EDBR for uniform movement; followed by two online heuristics for arbitrary movement in Sec. V. The evaluation of these heuristics is presented in Sec. VI and related work is discussed in Sec. VII. Finally, the paper is concluded in Sec. VIII.

II. System model

Our system model consists of mobile objects (MOs) and a location manager (LM) that maintains the positions of MOs. We do not make any assumption on the internal organization of the LM. It might comprise multiple LM nodes, to which the MOs are mapped (dynamically), cf. [8]. Each MO reports its position information to a single LM node over a wireless network.

An MO represents any mobile device (like cell phone or PDA) that is equipped with a processor, a wireless network interface and a positioning sensor to detect its own geographic position. In these devices, energy is a valuable and limited resource. To supply the LM with current position information, an MO has to perform three different operations: processing, position sensing and communication. We focus on the last two operations because they dominate energy consumption [2],[14].

Communication: An MO sends position update messages to the LM according to the underlying reporting protocol. As all update messages will be similar in size, we assume this transmission requires a constant amount of energy W_U per message.

Sensing: For sensing operations we use a generic model, applicable to a broad class of positioning sensors, though derived from current GPS technology [11]: Position sensing is not be performed continuously to conserve precious energy. Instead, the positioning sensor determines its current location by performing a *position fix*. Each position fix is explicitly invoked by the processor and requires some amount of time T_{sense} before the position is obtained. For example, GPS needs about $0.5\,\mathrm{s}$ for pseudo-range measurements of satellite signals and computing a valid position [11]. Each position fix also requires a constant amount of energy W_S. In between two fixes the positioning sensor can operate in a low-power *sleep* mode.

Note that we do not consider any background energy that is not influenced by these two operations. Its consumption is independent of the reporting protocol. E.g., a GPS receiver might still wake up *periodically* to keep a lock on the satellite signals.

The position of an MO at any time t is denoted $\vec{p}(t)$. For clarity of presentation, we assume that the position information obtained by the sensors is accurate. It has been shown (e.g., [7],[13]) how to cope with limited sensing accuracy in distance-based reporting. Furthermore, we assume that each MO has knowledge about its maximal velocity, denoted by v_{max}. It does not need to be a tight bound. This assumption is common for reporting protocols, and determining reasonable values has been discussed elsewhere, e.g., [7],[13].

Finally, we do not assume movements to be limited to a road network. Thus, we use the Euclidean distance to measure how far two positions are apart. However, our approach could be easily extended to support network-constrained movements (cf. [1],[16]) by using the travelling distance instead.

III. Basic Idea: Early Distance-Based Reporting

In this section, we first analyze the energy cost of *Distance-based Reporting (DBR).* This reveals a fundamental trade-off between the energy consumption of sensing and update operations. Motivated by this insight, we then present our basic idea to save energy by sending position updates earlier.

As mentioned before, DBR allows applications to select the required accuracy of location information. The selected accuracy determines the *update threshold* d_{th}, that must not be exceeded in between two position updates. That is, an MO must always send a new update message before its distance to the previous reported position reaches d_{th}. Let $t_{i,0}$ and $t_{i+1,0}$ denote the time of the i-th and i+1-th position update, respectively. Then, DBR fulfills the following condition:

$$\forall t \in [t_{i,0},\ t_{i+1,0}]:\ \left|\vec{p}(t) - \vec{p}(t_{i,0})\right| \le d_{\text{th}} \tag{1}$$

To minimize the energy spent on communication, the MO typically sends its updates as late as possible. This is particularly efficient for slow or sporadic movements, because the object can spend a long time without sending any update.

At the same time, the MO must locally monitor its position, to detect reaching the update threshold. Instead of continuous sensing, it can use a technique called *selective sensing* to conserve more energy: After each position fix the MO computes the time it can suspend sensing without violating condition (1). Obviously, this is the minimum amount of time required to reach the update threshold based on the MO's maximal velocity. Therefore, after sending a position update the MO can defer the next position fix for $d_{\text{th}} / v_{\max}$. If the update threshold is not yet reached after that time, the next fix can be scheduled based on the remaining distance to d_{th}.

In detail, let $t_{i,j}$ denote the time the j-th position fix, that follows the i-th position update, is completed (cf. Fig. 1). So, $t_{i,0}$ is the time of the position fix preceding the i-th update, $t_{i,1}$ is the time of the first position after the i-th update is obtained, and so on. At time $t_{i,j}$ the MO then computes the waiting time $T_{\text{wait}}(t_{i,j})$ for the next position fix as follows:

$$d_{\text{dist}}(t_{i,j}) = \left|\vec{p}(t_{i,j}) - \vec{p}(t_{i,0})\right| \tag{2}$$

$$T_{\text{wait}}(t_{i,j}) = \frac{d_{\text{th}} - d_{\text{dist}}(t_{i,j})}{v_{\max}} \tag{3}$$

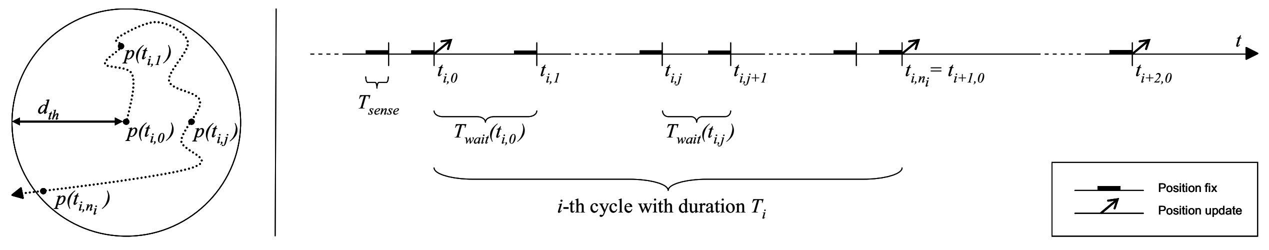

Figure 1. Timeline of sensing and update operations in distance-based reporting.

After computing $T_{\mathrm{wait}}(t_{i,j})$, the MO has to decide whether or not to send a position update to the LM. Due to (1), the update must be performed if the next position fix is not known to be completed before d_{th} is reached. This corresponds to the so-called D*istance-based Update Condition (DBU-condition)*:

$$T_{\mathrm{wait}}(t_{i,j}) < T_{\mathrm{sense}} \tag{4}$$

Recall that T_{sense} is the time required by the position sensor to obtain the position. If the DBU-condition is fulfilled, a position update is sent to the LM and the waiting time is recomputed accordingly (using $t_{i+1,0} = t_{i,j}$). In either case, the MO then waits $T_{\mathrm{wait}}(t_{i,j}) - T_{\mathrm{sense}}$ before it initiates the next positing fix. During that time, the positioning sensor can stay in sleep mode.

Consequently, an MO performs a series of position fixes, which are divided into so-called *cycles*. Each position update starts a new cycle and terminates the old one. That is, the i-th cycle comprises the sequence of position fixes completing at $t_{i,1}, t_{i,2}, \ldots, t_{i+1,0}$. We again refer to Fig. 1.

DBR has been designed for minimizing the number of position updates, while meeting the required accuracy. But, by considering the power consumption of this protocol, we can see that updating as late as possible is not always the best strategy to save energy. Let W_i denote the energy consumed during the i-th cycle. Further, T_i and n_i denote the duration and the number of position fixes of that cycle, respectively. Then, the average power consumption $\overline{P}$ results from the weighted average of each cycle:

$$\overline{P} = \frac{\sum W_i}{\sum T_i} \quad \text{with} \quad W_i = W_{\mathrm{U}} + n_i \cdot W_{\mathrm{S}}$$

$$\text{and} \quad T_i = \sum_{k=0}^{n_i - 1} T_{\mathrm{wait}}(t_{i,k}) \tag{5}$$

This shows that the factor n_i / T_i has a critical impact on the resulting power consumption. The problem with DBR is that the frequency of position fixes increases the closer the MO comes to the update threshold. This is because the waiting time between two fixes decreases the closer the threshold (see (3)). Furthermore, a short waiting time does not allow the MO to cover a large distance before the next position fix is performed. Thus, it is still located relatively close to the update threshold after that fix. As a consequence, the power consumption may increase substantially while the MO approaches the threshold.

In order to avoid this negative effect, updates could be sent earlier. However, this decreases sensing costs by increasing update costs. Thus, both costs must be balanced carefully to reduce the total energy consumption. This motivates the *Early Distance-based Reporting (EDBR)* protocol, we propose. Its main algorithm is depicted in Fig. 2. On this level of abstraction, the only difference to DBR is that we added a new condition, called *Energy-based Update Condition (EBU-condition).* This is used in addition to the DBU-condition (see line 6) to trigger a new update whenever this is advantageous with regard to energy consumption.

The interesting question is how to realize this EBU-condition. It should be easy to evaluate at runtime and conserve as much energy as possible. Unfortunately, an optimal solution can only be achieved if the future movement of the MO is

```
Main:
<1>    while(reporting) do {
<2>         newPos := readSensor();          //acquire new position
<3>         d_wait := d_th - dist(lastUpd, newPos);
<4>         T_wait := d_wait / v_max;
<5>
<6>         if (DBU_condition || EBU_condition){
<7>              sendUpdate(newPos);          // send msg.  to LS
<8>              lastUpd := newPos;
<9>              T_wait := d_th / v_max;      // reset waiting time
<10>        }
<11>        sleep(T_wait - T_sense);         //low-power mode
<12>   }

DBU_condition:
<13>   return (T_wait < T_sense)
```

Figure 2. Main algorithm of Early Distance-based Reporting (EDBR).

known in advance. The development of an appropriate EBU-condition is subject of the following two sections.

IV. Energy-based Update Condition for Uniform Movement

The object's movement can affect the resulting energy consumption of EDBR significantly. Here, we analyze one particular, simple movement pattern, called *uniform movement*: an MO is assumed to move linearly with constant speed $|\vec{v}| \le v_{\max}$. I.e., its position at some time can be determined by a linear function: $\vec{p}(t) = \vec{p}_0 + \vec{v} \cdot t$. For this movement pattern, we analytically derive an EBU-condition that is optimal in terms of energy cost. Although only few objects will move uniformly in reality, the analytical model gives valuable insights into the problem. More importantly, we utilize these results for the heuristics introduced in the next section.

With uniform movement, the MO always covers a constant distance per time. Let $T_{i,j}$ denote the time between sending the update starting cycle i and the j-th position fix in cycle i. That is, $T_{i,j} = t_{i,j} - t_{i,0}$. Then, the distance covered in that time is $d_{\mathrm{dist}}(t_{i,j}) = |\vec{v}| \cdot T_{i,j}$. Using (3) we can determine $T_{i,j}$ recursively:

$$T_{i,0} = 0$$

$$T_{i,j} = T_{i,j-1} + T_{\mathrm{wait}}(t_{i,j-1}) = \frac{d_{\mathrm{th}}}{v_{\max}} + \left(1 - \frac{|\vec{v}|}{v_{\max}}\right) \cdot T_{i,j-1}$$

This polynomial sequence also has a closed form (which can be shown by complete induction):

$$T_{i,j} = \begin{cases} j \cdot \dfrac{d_{\mathrm{th}}}{v_{\max}} & , |\vec{v}| = 0 \\ \dfrac{d_{\mathrm{th}}}{|\vec{v}|} \cdot \left(1 - \left(1 - \dfrac{|\vec{v}|}{v_{\max}}\right)^{j}\right) & , |\vec{v}| > 0 \end{cases} \tag{6}$$

Next, we can determine the average power consumption, denoted as $\overline{P}_{\mathrm{lin}}$, based on the following observation: With uniform movement, each cycle of EDBR will have the same "temporal structure" in terms of position fixes. That is, after j fixes within a cycle, the MO is always located at the same distance to the last updated position ($\forall k, l : d_{\mathrm{dist}}(t_{k,j}) = d_{\mathrm{dist}}(t_{l,j})$). Any deterministic update condition will thus yield the same amount of position fixes during each cycle. Consequently, P_{lin} is equal to the power consumption of one cycle and depends only on the number of position fixes per cycle, say n. Using (5), this yields

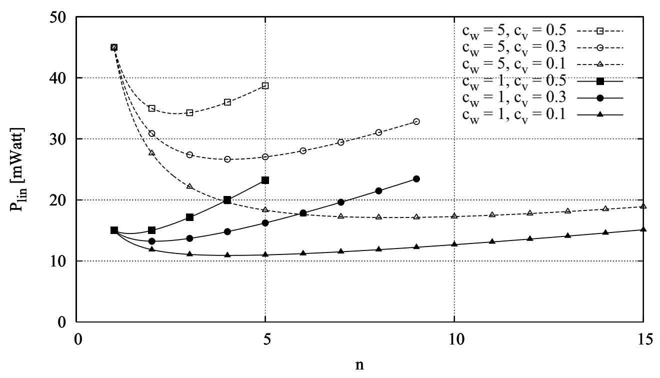

Figure 4. Average power consumption $P_{\text{lin}}(n)$ for different $n \leq n_{\max}$.

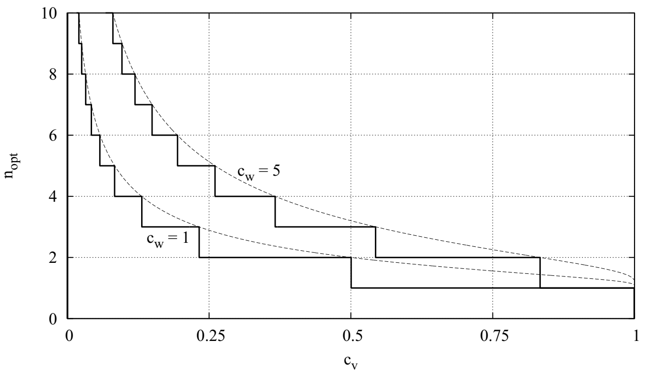

Figure 3. n_{opt} for varying values of c_v and c_w.

$$\overline{P}_{\text{lin}}(n) = \frac{W_U + n \cdot W_S}{T_{i,n}} . \quad (7)$$

In the following, let us first consider the case $|\vec{v}| = 0$. Obviously, this reflects a non-moving object with $\vec{p}(t) = \vec{p}_0$. Hence, the same waiting time is obtained after each position fix, namely $T_{\text{wait}}(t_{i,0})$. This yields $T_{i,n} = n \cdot T_{\text{wait}}(t_{i,0})$. Accordingly, the resulting power consumption (7) always decreases for larger n. Unsurprisingly, this affirms that a non-moving object should not send any further updates (beyond the first) to save most energy. In doing so, the power consumption results in:

$$\overline{P}_{\text{lin}(v=0)} = \lim_{n\to\infty} \overline{P}_{\text{lin}(v=0)}(n) = \frac{v_{\max}}{d_{\text{th}}} \cdot W_S . \quad (8)$$

For any $|\vec{v}| > 0$, however, (6) and (7) yield an average power consumption of

$$\overline{P}_{\text{lin}}(n) = \frac{|\vec{v}|}{d_{\text{th}}} \cdot \frac{W_U + n \cdot W_S}{1 - \left(1 - \frac{|\vec{v}|}{v_{\max}}\right)^n} \quad ,1 \leq n \leq n_{\max} \quad (9)$$

where $n_{\max}$ is the maximal number of fixes that can be performed before the DBU-condition enforces another update. According to (4), this requires

$$\forall j < n_{\max} : T_{\text{wait}}(t_{i,j}) = \; = T_{i,j+1} - T_{i,j} \geq T_{\text{sense}} . \quad (10)$$

Using (6) this can be resolved to:

$$n_{\max} = \left\lfloor \log_{\left(1 - \frac{|\vec{v}|}{v_{\max}}\right)} \left(\frac{v_{\max} \cdot T_{\text{sense}}}{d_{\text{th}}} \right) + 1 \right\rfloor . \quad (11)$$

The resulting energy consumption $\overline{P}_{\text{lin}}(n)$ is a convex function with one global minimum. This is illustrated in Fig. 3 for different ratios of $c_V := |\vec{v}| / v_{\max}$ and $c_W := W_U / W_S$ [1]. Basically, all curves show a very similar behavior: At first, the average power consumption decreases with a higher number of position fixes. This is due to the fact that the costs for sending updates amortize over a larger period of time. Yet, after a certain number of fixes, the average power consumption starts to increase again, because the waiting time in between two fixes shortens as the MO approaches the update threshold.

As a consequence, the lowest power consumption is achieved, if the EBU-condition triggers an update after the last position fix that shows decreasing power consumption. That is, the EBU-condition must evaluate to true after n_{opt} fixes, with $n_{\text{opt}} := \max\{n \in \mathrm{N}^+ \mid \overline{P}_{\text{lin}}(n-1) \geq \overline{P}_{\text{lin}}(n)\}$. Using (9), and c_v, c_w as previously defined, this results in:[2]

$$n_{\text{opt}} = \max\left\{ n \in \mathrm{N}^+ \mid \frac{c_W + n - 1}{1 - (1 - c_v)^{n-1}} \geq \frac{c_W + n}{1 - (1 - c_v)^n} \right\}$$

$$= \left\lfloor \frac{1}{\ln(1 - c_V)} \cdot \Omega\left(\frac{\ln(1 - c_V)}{c_v} (1 - c_V)^{\frac{1}{c_V} + c_W} \right) - \frac{1}{c_V} - c_W + 1 \right\rfloor . \quad (12)$$

Though this is a complex function, n_{opt} can be determined efficiently by iterative computation of $\overline{P}_{\text{lin}}(n+1)$ until it greater than $\overline{P}_{\text{lin}}(n)$ for the first time. Fig. 4 illustrates the resulting n_{opt} for different values of c_v and c_w. Note that n_{opt} depends on these two ratios only, rather than the update threshold d_{th}. The reason is that an MO with uniform movement covers a constant fraction of the remaining distance in between two position fixes.

Next, it can be observed that n_{opt} decreases with larger values of c_v (with a higher velocity). In that case, the MO can perform less position fixes before reaching d_{th}. Thus, the waiting times decrease more rapidly, which causes higher power consumption with the same amount of position fixes (cf. Fig 3). But interestingly, the distance covered after n_{opt} position fixes nevertheless increases with larger values of c_v. For example, $n_{\text{opt}} = 8$ is obtained for $c_w = 5$, $c_V = 0.1$, which corresponds to a distance of 57 m. Whereas $c_V = 0.3$ yields $n_{\text{opt}} = 4$ but a distance of 76 m.

Furthermore, n_{opt} also decreases with smaller values of c_w. This is because updating operations have a less dominating impact on the energy consumption with decreasing c_w. As a consequence, an early update can be amortized by longer waiting times more easily.

Finally, we have to point out that both n_{opt} and the resulting power consumption $\overline{P}_{\text{lin}}(n_{\text{opt}})$ are significantly lower than $n_{\max}$ and $\overline{P}_{\text{lin}}(n_{\max})$ respectively (cf. Fig. 3). Resuming the previous example, $n_{\text{opt}} = 8$ yields a power consumption of 17.1 mWatt, opposed to 26.8 mWatt for $n_{\max} = 29$. This constitutes energy savings of more than 36% compared to the original DBR.

[1] Throughout this section, we consistently use the following sample values to illustrate the results: $T_{\text{sense}} = 0.5$ s, $d_{\text{th}} = 100$ m, $v_{\max} = 10$ m/s, $v = c_V \cdot v_{\max}$, $W_S = 75$ mJoule, and $W_U = c_W \cdot W_S$.

[2] The deviation of this function is omitted due to limitations in space but can be provided upon request. Note that $\Omega(w)$ denotes the Lambert W function. It is the inverse of $f(w) = w \cdot e^w$.

V. Energy-based Update Condition for Arbitrary Movement

In this section, we discuss how to apply these findings to a real-world setting. Now, we consider an arbitrary movement of the MO that is not known in advance. We present two online heuristics that evaluate the EBU-condition at runtime. Basically, both heuristics follow the same rationale. They first assess the resulting power consumption of both alternatives – sending an update immediately or scheduling the next position fix only. Then, the option that requires less energy in the future is chosen. Without knowing the future movement, however, the resulting power consumption can only be estimated. The two heuristics differ in how this is tackled. The *Next Fix Heuristic* considers the respective power consumption of the very next waiting time. The *Predicted Movement Heuristic* uses past position information to predict the MO's future movement as a linear function. In the following we will discuss both heuristics in detail.

A. Next Fix (XF) Heuristic

This update strategy utilizes the fact that the behavior of EDBR up to the next position fix is known in advance. At the time the EBU-condition is evaluated, say $t_{i,j}$, the MO can already determine the waiting time for both, the update and the no-update option. Thus, it can easily compare the power consumption of both options up to the time of the next position fix:

With the no-update option, the MO waits for a period of $T_{\text{wait}}(t_{i,j})$ at the cost of one position fix. With (3), the power consumption during that time is:

$$\overline{P}_{\text{no-upd}} = \frac{W_{\text{S}}}{T_{\text{wait}}(t_{i,j})} = \frac{v_{\max}}{d_{th} - d_{\text{dist}}(t_{i,j})} \cdot W_{\text{S}} \tag{13}$$

With the update option, the MO spends additional energy for a position update but benefits from a longer waiting time $T_{\text{wait}}(t_{i+1,0})$ at the beginning of a new cycle. Using (3) again, this yields in:

$$\overline{P}_{\text{upd}} = \frac{W_{\text{U}} + W_{\text{S}}}{T_{\text{wait}}(t_{i+1,0})} = \frac{v_{\max}}{d_{\text{th}}} \cdot (W_{\text{U}} + W_{\text{S}}) \tag{14}$$

Obviously, an update should be sent only if $\overline{P}_{\text{upd}} < \overline{P}_{\text{no-upd}}$. Resolving this inequation leads to the following EBU-condition of *EDBR-XF*:

$$d_{\text{dist}}(t_{i,j}) > \frac{W_{\text{U}}}{(W_{\text{U}} + W_{\text{S}})} \cdot d_{\text{th}} \tag{15}$$

Consequently, the MO should send a new position update when the distance to the last reported position $d_{\text{dist}}(t_{i,j})$ exceeds a certain fraction of the update threshold. This fraction can be determined in advance provided that the energy costs of sensing and update operations are constant. For example, consider an energy ratio of c_w=3. Then, this heuristic advises a new update whenever ¾ of the allowed update distance (d_{th}) are exceeded after a position fix.

In fact, this constitutes a distinct advantage of the XF heuristic. It imposes virtually no extra runtime overhead and thus can be implemented even on devices with very limited computing capabilities. Furthermore, this heuristic provides a guaranteed upper bound on the total power consumption. Since an update is sent whenever $\overline{P}_{\text{upd}} < \overline{P}_{\text{no-upd}}$, the power consumption during each waiting time is known to be less than or equal to $\overline{P}_{\text{upd}}$. The same holds for the average of all waiting times, too.

On the downside, a drawback of the XF heuristic is that it does not consider the MO's situation after the next position fix. However, this situation may be different for the two options. If no update is sent, there is a high probability that the object is still closer to the update threshold after the next position fix, which again causes a shorter waiting time for the following fix. On the other hand, sending an update instead often prolongs not only the next but also following waiting times. As a consequence, XF tends to penalize the update option. This effect increases for lower speeds because more position fixes have to be performed while a certain distance is crossed.

B. Predicted Movement (PM) Heuristic

In contrast to XF, the PM heuristic considers the power consumption *beyond* the next position fix. Its EBU-condition is based on a prediction of the MO's further movement. We apply a simple prediction function that assumes uniform movement, which allows us to build on the results presented in Sec. IV. PM always chooses the option that requires less energy for that anticipated movement.

After each position fix, say at time $t_{i,j}$, PM first predicts the future movement of the MO by means of a linear prediction function. This assumes the MO continues its current movement uniformly – without changing speed or direction. In particular, the predicted movement vector is obtained by linear extrapolation of the last two positions: $\vec{v}_{\text{p}} = \vec{p}(t_{i,j}) - \vec{p}(t_{i,j-1})$. Note that more than two preceding position fixes could be used as well in order to smooth out sensing errors. However, this also determines how quickly PM reacts to changes in the movement.

Based on the predicted uniform movement, PM then compares the average power consumption of the update and no-update option in the future. Let us first consider the update option. Sending an update will start a new cycle. In order to anticipate the power consumption for that new cycle we can directly apply equations (9) and (12). Thus, the power consumption for the new and all following cycles amounts to $\overline{P}_{\text{lin}}(n_{\text{opt}})$, where n_{opt} denotes the optimal length of each cycle (cf. (12)). That is, the minimal power consumption for the update option is:

$$\overline{P}_{\text{upd}} = \overline{P}_{\text{lin}}(n_{\text{opt}}) \tag{16}$$

With the no-update option, a position update is not sent immediately but only after $k > 0$ further position fixes. In other words, the current cycle continues and a new cycle is started after k more position fixes. Let $\overline{P}_{\text{no-upd}}(k)$ denote the average power consumption for the remaining portion of the current cycle. Then, it is important to notice that the average power consumption of all following cycles amounts to $\overline{P}_{\text{lin}}(n_{\text{opt}})$ again. This is depicted in Fig. 5, which shows the update option and the no-update option for k=1 and k=2. Consequently, PM chooses the update option only if there exist no $k > 0$ with $\overline{P}_{\text{no-upd}}(k)$ less than $\overline{P}_{\text{upd}}$. This amounts to the following EBU-condition of *EDBR-PM*:

$$\overline{P}_{\text{lin}}(n_{\text{opt}}) < \min\left(\overline{P}_{\text{no-upd}}(k)\right), \quad k > 0 \tag{17}$$

As pointed out in the previous section, $\overline{P}_{\text{lin}}(n_{\text{opt}})$ can be determined efficiently by means of an iterative computation. $\overline{P}_{\text{no-upd}}(k)$ differs from $\overline{P}_{\text{lin}}(n_{\text{opt}})$ since it does not comprise a

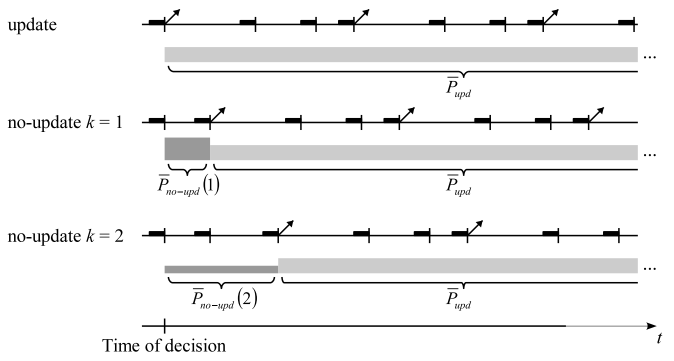

Figure 5. Comparing the power consumption of different options.

complete cycle. However, it can be shown that its minimum can be computed in an analogous manner since $\overline{P}_{\text{no-upd}}(k)$ is also a convex function with one global minimum. Note that the minimum is actually located at $k = 1$, whenever the predicted movement yields $d_{\text{dist}}(t_{i,j}) < d_{\text{dist}}(t_{i,j+1})$.

Only in the special case of $\left|\vec{v}_{\text{pred}}\right| = 0$, the minimum is obtained for $k \to \infty$. In that case, all waiting times before sending an update will be equal to $T_{\text{wait}}(t_{\text{i,j}})$ and analogous to (8) the resulting power consumption is: $W_{\text{S}} / T_{\text{wait}}(t_{\text{i,j}})$. With (3) and (8), the EBU-condition (17) then reduces to $d_{\text{dist}}(t_{\text{i,j}}) > 0$ in that particular case.

Finally, the resulting algorithm to determine EBU-condition of EDBR-PM is depicted in Fig. 6. In line 2 it first handles $\left|\vec{v}_{\text{pred}}\right| = 0$ explicitly. For any other predicted velocity both $\min(\overline{P}_{\text{no-upd}})$ and $\overline{P}_{\text{lin}}(n_{\text{opt}})$ are determined by iteratively computing *P(i+1)* until *P(i) < P(i+1)* for the first time (line 4 and 5 respectively). Note that the computation of $\overline{P}_{\text{lin}}(n_{\text{opt}})$ can be terminated even earlier, as soon as its value becomes smaller than $\min(\overline{P}_{\text{no-upd}})$ (cf. line 17).

Usually, the number of iterations required is quite low (as shown in Fig. 4). Therefore, the EBU-condition of PM can be evaluated efficiently, although the overhead is higher than for the XF heuristic. A clear advantage of PM is that the update condition is adapted to the predicted movement. Although we have chosen a rather simple prediction function, our evaluations show performance improvements compared to XF. Potential prediction errors have limited effect since the movement vector is corrected with every position fix.

```
EBU_condition:
<1>     mov := linearMovementPrediction(...);
<2>     if (mov.velocity == 0) return (dist(lastUpd, newPos) > 0);
<3>
<4>     P_wait := predictP(lastUpd, mov, WS , T_wait , 0 );
<5>     P_lin := predictP(newPos, mov, WS+WU, d_th / vmax, P_wait);
<6>     return (P_lin < P_wait);

PredictP (Position up, Prediction mov,
          Energy W, Duration T, Power P_low):
<7>
<8>     /* next waiting time */
<9>     pos := mov.posAfter(T);
<10>    d_wait := d_th - dist(up, pos);
<11>    T_wait := d_wait / v_max;
<12>
<13>    /* compare power consumption */
<14>    P_last := W/T; W += WS; T += T_wait; P_new := W/T;
<15>    if (T_wait < T_sense) return P_last:       // DB-update
<16>    if (P_new > P_last) return P_last;         // power increases
<17>    if (P_new < P_low) return P_new;           // already lower
<18>
<19>    /* else: perform next iteration */
<20>    return predictP(up, mov, W, T, P_low);
```

Figure 6. EBU-condition for the Predicted Movement (PM) heuristic.

VI. Evaluation

To evaluate the performance of our heuristics, we report on various experiments based on a simulation of walking pedestrians. In particular, we compare EDBR-XF and EDBR-PM with the original DBR protocol in terms of power consumption. Next, we explain the experimental setup, followed by the detailed results.

A. Simulation Setup

We used the CanuMobiSim simulator [15] to generate movement traces of pedestrians following random trip sequences through a typical European city. This scenario comprises a simulation area of 2.0 x 2.0 km². The speed of movement is chosen randomly from 0-3 m/s every 30 seconds. Each movement trace captures a period of 3 hours. All depicted results represent the average of 500 simulation runs with different movement traces.

The protocols assume a maximal velocity of $v_{\text{max}} = 10$ m/s, which reflects a pessimistic bound on actual speeds. Concerning energy costs, we assume that sending an update message consumes $W_{\text{U}} = 150$ mJoules. According to [3], this amount suffices to transmit about 240 bytes over GSM/GPRS. Each position fix is assumed to cost $W_{\text{S}} = 75$ mJoules and to take $T_{\text{sense}} = 0.5$ s. These are typical values of a low-power GPS receiver [12]. Together, this yields an energy ratio of $c_{\text{W}} = 2$.

B. Sensing and Update Operations

In the first set of experiments we measured the performance of all protocols for varying update thresholds d_{th}. Fig. 7 presents the amount of (a) position fixes and (b) position updates that are performed by one MO per hour, as well as (c) the resulting power consumption.

First, it can be observed that all protocols show a similar performance for small values of d_{th}. This is due to the fact that the DBU-condition enforces a position update after almost every fix. Recall that a new report is required if the remaining distance to d_{th} is less than $v_{\text{max}} \cdot T_{\text{sense}} = 5$m, cf. (4). Thus, with an update threshold of only 5 m the MO has to send an update whenever it changes its position in between two fixes. With increasing d_{th}, T_{sense} becomes less influencing since the position must be reported with a lower accuracy only. Hence, the MO can move for a longer period of time without update.

For larger d_{th}, DBR shows the highest decrease of position updates (cf. Fig. 7a), because it sends an update only when reaching the update threshold. As expected by design, a higher amount of updates is observed for EDBR. But it can also be seen that EDBR-PM generates even more updates than EDBR-XF. That is, on average the PM heuristic generates a new update earlier (after less position fixes) than XF.

Concerning the amount of position fixes (Fig. 7b) we can observe a converse relation. EDBR-PM performs even less position fixes than EDBR-XF. At the same time, both variants of EDBR save a significant amount of position fixes by early reporting compared to DBR. DBR suffers from very short waiting-times whenever the MO is close to the update threshold, which results in the highest amount of position fixes.

The overall power consumption, including the costs of both sensing and updating, is depicted in Fig. 7c. Here, the benefit of our solution becomes apparent. It can be seen that DBR always experiences the highest power consumption due to the

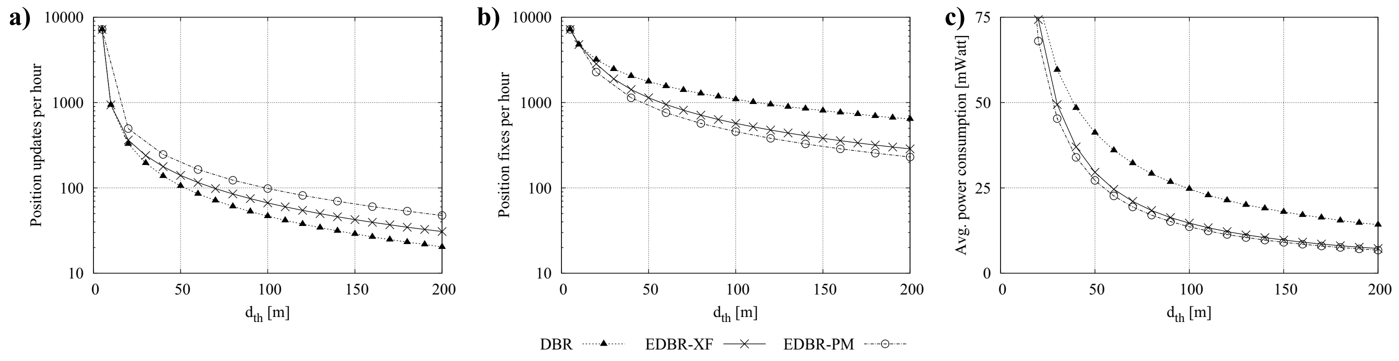

Figure 7. a) Number of position updates per hour, b) Number of positon fixes per hour, c) Resulting average power consumption.

large amount of position fixes. In contrast, both EDBR variants always consume less energy. For example, with $d_{th} = 50$ m EDBR-XF saves 28% of the energy consumed by DBR, and EDBR-PM even saves 34%. Moreover, the energy savings further increase with larger values of d_{th}. With $d_{th} = 200$ m they amount to 49% and 52% respectively. The reason is that the ratio of very short to long waiting times increases with d_{th}. This amplifies the gain of sending an update earlier, because a larger fraction of position fixes is saved (cf. Fig. 7b).

Finally, Fig. 7c also shows that the PM heuristic performs better than XF in terms of power consumption for all values of d_{th}. Thus, PM benefits from the movement prediction that allows adapting the EBU-condition to different movement characteristics. In contrast, the EBU-condition of XF is more conservative in that it generates less position updates (cf. Fig. 7a), which causes a slight increase in power consumption only.

C. *Impact of Energy Costs and Movement Speed*

Next, we present further experiments to evaluate the impact of the ratios c_W and c_V on the power consumption. They are based on a fixed update threshold $d_{th} = 100$ m.

Since other technologies for sensing or communication are likely to yield different ratios of energy costs, we vary c_w by gradually increasing W_U. As a consequence, the impact of update operations becomes more dominating.

Fig. 8a depicts the resulting power consumption of all three protocols. It shows how the energy savings of both EDBR variants decrease with larger values of c_w. However, their power consumption always remains lower than DBR. This is due to the fact that their EBU-conditions take the energy costs into account. For higher update costs, new messages are sent less frequently. Thereby, the energy savings decrease accordingly. Nevertheless, with $c_w = 9$ the savings still amount to 12% for EDBR-XF and 21% for EDBR-PM. Furthermore, it can be observed that the improvement of EDBR-PM over EDBR-XF even increases with c_w. This shows that the predictive EBU-condition assesses the future power consumption more accurately. As explained in the previous section, XF tends to penalize the update-option and this effect increases with c_w, cf. (15).

Next, we evaluate the impact of different movement speeds in relation to the maximal velocity v_{max} (= 10 m/s). Therefore, movement traces are generated as previously described, but with a constant speed v. From one experiment to the next, v is gradually increased. The assumed energy ratio is $c_W = 2$ again.

The resulting power consumption in relation to c_V is shown in Fig. 8b. First, note that all protocols consistently show the lowest power consumption for $c_V = 0$. In that special case, the object does not move at all and hence no update messages are sent by any of the protocols. With increasing c_V more update messages are required because the MO approaches the update threshold faster. For $c_V = 1$ the MO always moves with v_{max}. Still, this does not necessarily require an update after each fix, due to changes in the direction of movement. Accordingly, DBR sends less updates than EDBR, but shows higher power consumption even at maximal speed.

Most importantly, it can be observed that the energy savings of EDBR over DBR are significantly higher for lower values of c_V. This clearly confirms the initial motivation of our work. When an object slowly approaches the update threshold, DBR causes a series of frequent position fixes. The benefit of sending an update earlier thus increases for lower speeds. Likewise, the improvement of PM over XF also increases with

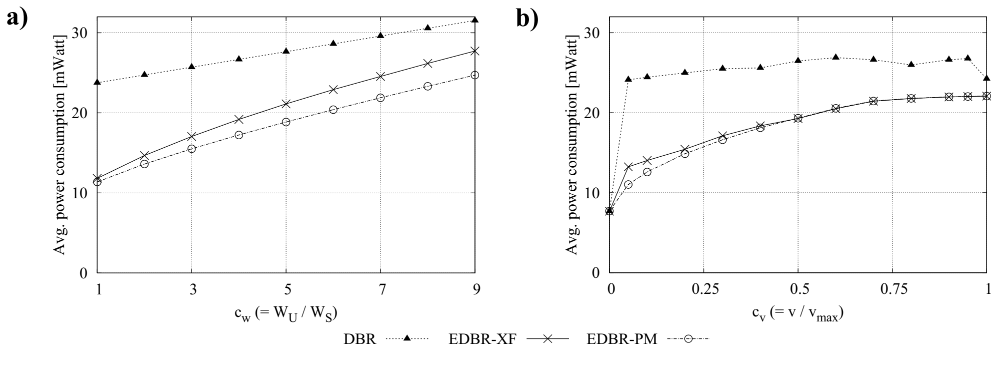

Figure 8. Average power consumption for diffenet ratios of a) energy costs c_w and b) movement speed c_v.

smaller c_V, because it sends the update even earlier. It shows that PM balances the costs of sensing and updating more evenly. In fact, for XF the effect of penalizing the update option (cf. Sec. V) increases with smaller speed ratios as well.

In summary, this evaluation shows that our solution of sending updates earlier can significantly decrease the energy consumption caused by both sensing and update operations. Beyond the movement scenario reported here, we also experimented with a variety of different mobility models and obtained comparable results. For example, with a Brownian movement, which frequently changes speed and direction, we observed energy savings over DBR of 48% for EDBR-PM and 36% for EDBR-XF respectively (with $d_{th} = 200$ m). Due to limitations in space we cannot present further results. Evaluating the accuracy of position information maintained by the LM is also omitted. We observed that the guaranteed accuracy (d_{th}) is never violated by neither DBR nor EDBR, as presented here. Moreover, the *average* accuracy of the remote data even improves with EDBR since updates are sent more frequently.

VII. Related Work

The work related to this paper can be divided into two categories, approaches to reduce sensing costs and protocols to report position information.

A good overview of strategies to reduce the *acquisitional* energy consumption in sensor networks is given in [14]. Commonly, these solutions exploit correlations between values of multiple sensors – either on the same node [2],[10], or on multiple nodes in spatial proximity [5]. The energy consumption is reduced by acquiring data from a subset of sensors only and predicting the expected value of others with some level of confidence. This differs from the problem considered in this paper.

Another approach to reduce sensing cost by *selective sampling* has been proposed for context-aware computing. In particular, [6] studies the trade-off between power consumption and prediction accuracy when using an *eWatch* with embedded accelerometer to predict the current situation of its user. However, none of these works addresses the inherent trade-off between sensing and updating, we are dealing with.

For tracking the position of MOs, a variety of reporting protocols have been proposed in the literature. A classification can be found in [7]. In [16] adaptive update policies are proposed based on an information cost model. Other approaches apply movement prediction functions [1],[9],[17]. Reducing the energy consumption has generally been an important design goal for such protocols. However, to the best of our knowledge, none of these protocols considers the impact of sensing operations.

It is important to notice that *prediction-based reporting protocols* can profit from considering sensing cost, too. With those schemes, each MO reports not only its current position but also some prediction of its movement. Thus, the next update is not required before the deviation between its real and predicted position exceeds a given threshold. This optimization of DBR still suffers from the same problem we studied in this paper: Whenever an MO approaches the update threshold, the frequency of position fixes increases in the same. We are currently studying how the early reporting approach we presented can be applied to prediction-based reporting protocols, too.

VIII. Conclusion

In this paper, we studied how valuable energy of MOs can be saved in distance-based reporting. We have shown that there exists an inherent trade-off between the energy consumption of both position sensing and updating. We proposed several strategies to reduce the resulting energy consumption by adapting the time of sending a new position update. For uniform movement the optimal update strategy was derived analytically. For arbitrary movement we developed two novel online-heuristics to evaluate the update decision at runtime. Although increasing the amount of position updates, they reduce the total energy consumption significantly. Our experiments indicate that more than 50% of the energy can be saved in a realistic scenario. Thus, we are convinced that our solution can provide a considerable improvement to the real-world deployment of location-based systems. In the future, we will extend this technique to prediction-based reporting and further explore the impact of position sensing on different update protocols.

References

[1] A. Civilis, C. Jensen, and S. Pakalnis: *"Techniques for Efficient Road-Network-Based Tracking of Moving Objects"*. In: IEEE Trans. Know. Data Eng. 17(5), May 2005.

[2] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong: *"Model-driven data acquisition in sensor networks"*. In: Proc. 30th Int'l Conf. Very Large Data Bases (VLDB'04), Toronto, Canada, Sep. 2004.

[3] B. Gedik and L. Liu: "*MobiEyes: A Distributed Location Monitoring Service Using Moving Location Queries*". In: IEEE Trans. Mobile Comp. 5(10), Oct. 2006.

[4] C. Jensen et al.: "*Location-based services - A database perspective*". In: Proc. 8th Scand. Conf. Geo. Inf. Science (ScanGIS'01), Ås, Norway, June 2001.

[5] Y. Kotidis: "*Snapshot Queries: Towards Data-Centric Sensor Networks*". In: Proc. 21st Int'l Conf. Data Engineering (ICDE'05), 2005.

[6] A. Krause et al.: "*Trading off Prediction Accuracy and Power Consumption for Context-Aware Wearable Computing*". In: Proc. 9th IEEE Int'l Symp. Wearable Computers (ISWC'05), Osaka, Japan, Oct. 2005.

[7] A. Leonhardi and K. Rothermel: *A Comparison of Protocols for Updating Location Information.* In: Cluster Computing – The Journal of Networks, Software Tools and Applications, 4(4), Oct. 2001.

[8] A. Leonhardi and K. Rothermel: *"Architecture of a Large-scale Location Service"*. In: Proc. 22nd Int'l Conf. Distr. Comp. Syst. (ICDCS'02), Vienna, Austria, July 2002.

[9] A. Leonhardi, C. Nicu, and K. Rothermel: *"A Map-based Dead-reckoning Protocol for Updating Location Information"*. In: Proc. Int'l Parallel & Distr. Processing Symp. (IPDPS'02), Ft. Lauderdale, USA, Apr. 2002.

[10] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. *"The design of an acquisitional query processor for sensor networks"*. In: Proc. ACM Int'l Conf. Mgmt. Data (SIGMOD'03), San Diego, USA, June 2003.

[11] P. Misra and P. Enge: *"Global Positioning System: Signals, Measurements and Performance"*, 2nd Ed., Ganga-Jumuna Press, 2006.

[12] Navman: *Jupiter 30 Data sheet*, http://www.navman.com/Documents/O-EM_docs/Jupiter 30/LA000576B_Jupiter30_DataSheet.pdf, Oct. 2006.

[13] D. Pfoser and C. Jensen: "*Capturing the Uncertainty of Moving-Object Representations*". In: Proc. 6th Int'l Symp. Spatial Databases (SSD'99), Hong Kong, China, July 1999.

[14] V. Raghunathan, S. Ganeriwal, and M. Srivastava: *"Emerging techniques for long lived wireless sensor networks"*. In: IEEE Comm. Mag. 44(4), Apr. 2006.

[15] I. Stepanov, P. Marron, and K. Rothermel: *"Mobility Modeling of Outdoor Scenarios for MANETs"*. In: Proc. 38th An. Simulation Symp. (ANSS'05), San Diego, USA, Apr. 2005.

[16] O. Wolfson, A. Sistla, S. Chamberlain, and Y. Yesha: *"Updating and Querying Databases that Track Mobile Units"*. In: Distributed and Parallel Databases, 7(3), July 1999.

[17] O. Wolfson and H. Yin: "*Accuracy and Resource Consumption in Tracking and Location Prediction*". In: Proc. 8th Int'l Symp. on Spatial and Temporal Databases (SSTD'03), Santorini, Greece, July 2003.