Objects Calling Home: Locating Objects Using Mobile Phones

Christian Frank[1], Philipp Bolliger[1], Christof Roduner[1], and Wolfgang Kellerer[2]

[1] Institute for Pervasive Computing, ETH Zurich, 8092 Zurich, Switzerland
{chfrank,bolligph,roduner}@inf.ethz.ch
[2] DoCoMo Communications Laboratories Europe, Munich, Germany
kellerer@docomolab-euro.com

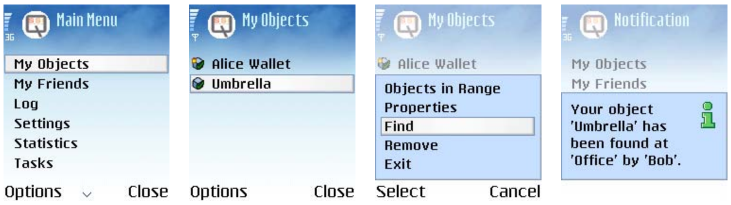
Abstract. Locating physical items is a highly relevant application addressed by numerous systems. Many of these systems share the drawback that costly infrastructure must be installed before a significant physical area can be covered, that is, before these systems may be used in practice. In this paper, we build on the ubiquitous infrastructure provided by the mobile phone network to design a wide-area system for locating objects. Sensor-equipped mobile phones, naturally omnipresent in populated environments, are the main elements of our system. They are used to distribute search queries and to report an object's location. We present the design of our object search system together with a set of simple heuristics which can be used for efficient object search. Moreover, such a system can only be successfully deployed if environment conditions (such as the participant density and their mobility) and system settings (such as number of queried sensors) allow to find an object quickly and efficiently. We therefore demonstrate the practicability of our system and obtain suitable system parameters for its execution in a series of simulations. Further, we use a real-world experiment to validate the obtained simulation results.

1 Introduction

To be able to locate everyday objects at the touch of a button is a promising application of ubiquitous computing. However, most systems which can be used for this purpose require an expensive infrastructure for sensing objects, for example, using Radio Frequency Identification (RFID) readers installed in the environment [1–3]. Their dependency on infrastructure precludes such systems from being used for locating objects in areas larger than confined indoor environments, particularly because of the costs involved in adding infrastructure to a significant fraction of the world, i.e., the space in which objects can be placed.

In this paper, we describe an object localization system based on mobile phones. Mobile phones combine two very useful features: They are omnipresent in environments in which users live and are at the same time inter-connected by a homogeneous world-wide infrastructure. Given that important objects can be augmented with an electronic tag such that they can be detected when brought into the vicinity of a mobile phone, many new applications become possible revolving around managing, monitoring, or locating one's everyday items.

Various technologies could be employed for object tagging and sensing objects within a short range of the mobile phone. For example, RFID tags are expected to be



(a) (b) (c) (d)

Fig. 1. User issues an object search query

attached to various consumer products in the near future as they may realize significant cost savings in stock and supply chain management. In particular, passive UHF RFID technology or active tags with a small autonomous power source [4] are expected to provide reading ranges of a couple of meters even with small reader modules. If improved variants of today's handheld RFID readers were integrated into mobile phones, a ubiquitous system could be deployed within a few years using the short innovation cycle established through mobile phone sales. In addition to RFID, other upcoming radio communication technologies, some even compatible with the phone's Bluetooth capability, could be used to identify objects in the phone's physical proximity in a similar way. If small, inexpensive Bluetooth-discoverable tags [5] can be built (in fact our prototype relies on battery-powered BTnodes [6]), a ubiquitous object sensing infrastructure is already in place today.

Note that each tagging technology defines a certain trade-off between tag costs, achievable identification range, and costs of reader hardware. Irrespective of the employed technology, we assume that *object sensors* can be integrated into mobile phones, as it has already been done with Bluetooth or NFC today. On a campus, in an office building, or, generally, any relatively dense urban environment, it would be possible to task mobile phones carried by other users with searching for an item one is interested in. In an office environment, any employee's mobile device could participate in sensing such tagged items and "call home", that is, notify the owner once the item is found using the short-range object sensor. If the notification includes some indicator on the location of the device which found the object, the owner may already have enough information to retrieve the item. For example, even a simple confirmation that some item has been left at work, as shown in Figure 1, can be very useful to users as they may stop searching somewhere else (e.g., at home) or just feel re-assured in case a valuable item is missing.

In this paper, we describe the design of our object sensing system based on mobile phones. A particular challenge of our application is to distribute an object search query to a subset of users (more generally, to object sensors) that are likely to find a given item. For this, we describe a set of heuristics which our system uses to define the scope of a query. Furthermore, by means of an extensive evaluation of the system's behavior, we demonstrate the practicability of the presented system and the used query scoping heuristics under varying operational conditions and with different system parameters. As a result of our evaluation, we obtain adequate system-parameter settings for a range of usage scenarios.

The remainder of the paper is organized as follows. We begin by surveying related work in Section 2 and describe the service architecture involved in providing our prototype's functionality in Section 3. In Section 4, we then discuss heuristics used for query scoping and detail the query dissemination protocol in Section 5. In Section 6, we provide initial evaluations obtained using our prototype in our own office environment. Further, in Section 7 we describe the simulation model we use to study the system's behavior in a wide-area environment and present the obtained results in Section 8. We summarize the results and present our conclusions in Section 9.

2 Related Work

Various related work has argued for the relevance of locating everyday objects, monitoring the presence of items, or avoiding their loss. Many such systems employ a pre-installed object sensing infrastructure [1–3]. Compared to these systems, we do not rely on a pre-installed infrastructure which is costly to deploy and to maintain, but focus on the use of mobile phones as hubs to a ubiquitous infrastructure. In the Smart Watch prototype [7], RFID readers transmit their current readings to the passing user's personal device, thus enhancing it with an object sensor. The user is then notified if objects are missing compared to readings which were collected earlier. Another prototype [8] allows train passengers to register their luggage with pre-installed object sensors, which then protect against theft and remind passengers of their items before unboarding the train. Both systems focus on reminding users before a loss takes place. As we assume that tagged objects will be numerous and often intentionally left behind, we aim to avoid immediate notification. Instead, we study how the location of an item can be determined on a user's request.

Note that an object search system could also be implemented by proactively sending all sensor readings to a centralized service which could be queried when an object needs to be located. Such a system would face the challenge of a global data collection system, such as IrisNet [9] or Hourglass [10]. In this paper, we use a reactive, that is, query-based approach, as the number of sensor readings (e.g., *object X seen by object sensor A*) is expected to be much larger than the number of queries. Moreover, our system avoids aggregation of all readings in a centralized database as this would have severe implications for the user's personal privacy. In particular, it is incompatible with a privacy enhancing feature of our system: Objects which have previously been associated to their owner, will only be detected by a sensor after this sensor has received an explicit query for the given object. Search queries for such objects contain the owner's key obscured by a random session identifier thereby implementing a lightweight authentication protocol [11] between owners and their objects.

Finally in [12], the authors use the mobile phone as a gateway to access heterogeneous health-related sensors which have been pre-installed in the environment of the user. Based on a different application, their system does not deal with two aspects which are central in this paper: Query scoping (determining which sensors should be queried from a large and homogeneous sensor array) and the obtained sensor coverage (identifying the numbers of users and the mobility patterns which allow for reliable detection of objects).

3 Use Case and Service Architecture

The main use case of our system, outlined in Figure 1, includes various aspects which we describe in this section together with the respective services implementing them.

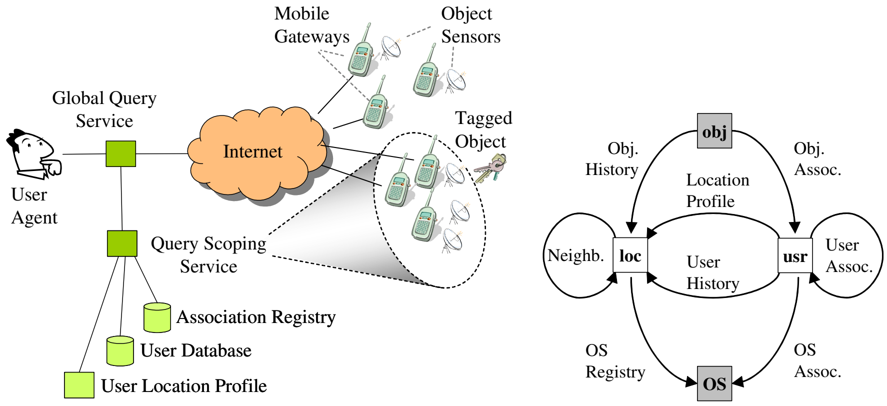
Association. The association service serves three main purposes. First, it keeps track of associations between users and objects (Figure 1(b)). Objects are visible to everybody in their initial state, but with association become visible only for queries initiated by their owner's device. Such association, accompanied by the exchange of a shared key [11], prevents other users from using the system to perform a wide-area search for the associated object. In this context, the mobile network operator may act as a gatekeeper and only distribute queries issued by object owners. Second, user to object sensor association maintains a set of object sensors which are particularly relevant to the user, for example, object sensors which have been installed at a remote holiday home. In the scenario of Figure 1, Bob's mobile device has been previously associated as a favorite object sensor. Third, similar user to user associations can be used to grant group access rights to certain objects, e.g., for families or groups of colleagues, or to determine in which circumstances users' identities are revealed.

Storage. However, our system does not depend on associated object sensors or users being near the tagged object at the time the query is issued. Instead, it stores certain context information when an associated object leaves the range of the local object sensor. Specifically, the mobile device stores a location trace of the user around the "loss" event which can help finding it later on. In a different usage scenario, which is not the focus of this paper, user-installed object sensors may simply report a stream of sensed objects, for example, carried by users passing by. For these cases, the service infrastructure provides users with a *user database* service that may be used as a sink for events generated by object sensors and mobile devices. Moreover, all association relationships are kept in a storage component called *association registry*. Storage services are available both on the user device and in the back-end infrastructure [13].

Localization. For remembering the location of an object when it goes out of range of the local object sensor and to provide location information for found objects, some location information must be available on the mobile device. While the prototype implementation is based on UMTS cell information, better localization functionality can be added in a future system if increased accuracy is desired. We will quantify the effect of varying positioning accuracy in Section 8.

Location Profile. In our prototype, we adapted [14] to perform statistics on the UMTS cells in which users spend most of their time. This will allow us to implement a search strategy which mainly considers locations where the user spends much time. Our prototype includes functionality for naming these locations [15], such as 'Office' in Figure 1(d).

Figure 2(a) gives an overview of the system architecture: As mentioned, the mobile phones are used to link *object sensors* to the back-end infrastructure. The back-end infrastructure hosts the *global query service*, which provides adequate dissemination support, cost control, and validity management for user queries, as we will describe in Section 5. The global query service in turn is based on the *query scoping service*, which implements application specific heuristics for retrieving the most appropriate subset of object sensors and is described in the following Section 4.



(a) System architecture (b) Data model

Fig. 2. System architecture and data model

4 Query Scoping Service

There are various heuristics for distributing a query to a relevant subset of object sensors. For example, one may distribute it to sensors near the location where the object was last in range of the local object sensor. Similarly, all conceivable heuristics will be based on some kind of history data available in the system. To elaborate on this, we show a simple data model of our application in Figure 2(b): *Objects* are associated with *users* (object owners) by the association service and also with *locations* (e.g., cells) in which an *object* has been observed in the past. Users may choose to record a history of their location on their mobile device (*user history*) or to enable the *location profile* service, which computes the *locations* that are most relevant for a given user (e.g., their home or office). In this simple model, locations are related to other locations via the *neighborhood* relation. Moreover, *users* can be associated with certain *object sensors* which they often use (e.g., which they have installed in their office or car) and with other *users* which are family, friends, or colleagues. Finally, the mobile network operator keeps a database (*OS registry*) which stores the current location (e.g., the current network cell) of certain mobile phones which can be used as *object sensors*.

Note that we omitted some details in the data model (most prominently a more refined location model). However, we can use the data model to show how many conceivable heuristics correspond to paths from an entity of type *obj* at the top to an entity of type *object sensor (OS)* at the bottom of Figure 2(b).

For example, we can query object sensors which:

I) Are near the location where the object was last seen.
II) Are near locations recently visited by the user.
III) Are near locations where the user spends a large amount of her/his time.
IV) Are associated with the object owner (as in Figure 1).
V) Match the above strategies III and IV for a different (associated) user, such as a family member, or even for a friend of a friend, etc.

While, intuitively, none of these heuristics can guarantee success, they all incorporate particular application-level assumptions on where users keep personal belongings and where these are generally left. Note how each heuristics represents a path in the data model of Figure 2(b): Heuristics I corresponds to the path (*obj-loc-OS*) on the left, while heuristics V corresponds to the path *obj-usr-usr-loc-OS*.

The application programmer may now define which relation types to use in the search by assigning weights to each one. The search algorithm, described in detail in complementary work [16], is then started with a source entity (i.e., the sought object) and a *destination type* (i.e., object sensors) as parameters. It will then "unfold" this annotated data model into a search tree of entities related to the sought object. Each edge in the tree, during algorithm execution, will be annotated with a relevance measure, derived from the user-defined relevance of the respective relation. The entities in the tree are visited in order of decreasing relatedness to the source entity. If a visited entity is of type *destination type*, it will be added to the algorithm's result list (in which contained entities are also ordered by their relatedness to the sought object). The algorithm stops once *entity limit* entities have been returned or relatedness falls below a given *relatedness threshold*.

The algorithm's advantage is that relations can be encapsulated in distributed components executed on various platforms of the system (e.g., some users would prefer to store their location profile on their mobile device only, while other relations are stored on the back-end server). In this paper, we do not focus on the search algorithm itself (details are described in [16]), but analyze the performance of the contained heuristics.

Please note that heuristics I-III employ the *OS registry* relation, which associates a set of locations L with a set of object sensors near them. Due to user mobility, however, the object sensors near the set L will change with time. To compute a query scope that is independent of user mobility, when heuristics I-III are implemented, *location* will be the *destination type* parameter passed to the search algorithm. The returned set of locations L, for example, a set of cells, is then passed on to the global query service, which will distribute the query to these locations as described in Section 5.

5 Global Query Service

If users are interested in finding an object o, they will issue a find query to the global query service. At this time, they may specify a message cost limit $q_{\max}$ denoting a limit on the messages sent during query dissemination and a time limit $t_{\max}$ after which the query will terminate at last.

If the search strategy IV is chosen, the set of object sensors is determined by the scoping algorithm. Here, a query will be distributed to the first $q_{\max}$ sensors returned by the algorithm and be active for at most $t_{\max}$ time.

If search strategies I-III are chosen, query scoping will not directly return a set of object sensors, but a set of locations. In the basic location model we employ, these can either be a set of cells (the most basic localization already available on the phone) or a set of geographic points (if phone localization is more precise) together with an associated measurement error. Because the set of object sensors associated with these locations may change over time, our system installs (or un-installs) a query at sensors which come close to (or, respectively, depart from) these locations. Whether a sensor

s is close to the returned locations is defined by the implementation of a predicate f (which maps s to either *true* or *false*).

Depending on the way *locations* are modeled, we use two different implementations of $f(s)$. Given a set of cells C, $f(s)$ will be true if the mobile phone (with its object sensor s) is currently served by any of the cells in C. Note that this information is already available at the mobile network operator, that is, it can be accessed without additional costs on the server side of our infrastructure.

In case the mobile devices are equipped with more accurate positioning means, the locations returned by query scoping will instead be a set of geographic points P. Here, $f(s)$ will be true if the current position measured by a mobile phone's object sensor s is within a certain range r away from the points P. This range r will depend on the error incurred at the positioning sensor when the points in P were measured (we will discuss a concrete implementation in our evaluation section). Note that such additional positioning information will only improve the efficiency of query dissemination if positioning information of all object sensors is already known at a database on the server. Otherwise, it would be inefficient to propagate all object sensor positions to the server before query dissemination, and therefore a different approach is chosen: The query is distributed to object sensors in a set of cells C which "cover" the whole area surrounding the points P (the actual object sensor will be turned on only later, once the predicate $f(s)$ evaluates to true). Note that the total number of distributed queries is now the same as if locations were a set of cells C.

When installing queries for such "location-based" strategies I-III, the total number of object sensors at which a query will be installed (q_{total}) is made up of two parts, $q_{\text{total}} = q_{\text{init}} + q_{\text{mob}}$. Here, q_{init}, denotes the number of users queried initially at the time the query is issued. At this time, q_{init} users/sensors are randomly chosen out of the initial query scope $S_{\text{init}} = \{s|f(s) = \text{true}\}$. The number q_{init} is set as

$$q_{\text{init}} = \min(q_{\max}, |S_{\text{init}}|, A/o_A) \tag{1}$$

where o_A represents the area which can be covered by a typical object sensor (for example a disk around the sensor with a given radius) while A denotes an estimate of the total area in which f would return true. Here, we assume that due to user mobility, A/o_A object sensors should be enough to cover the total area involved, although if all users were stationary, sensing areas o_A may overlap and more queries would be required.

In addition to q_{init}, the query will be installed at a second set of sensors q_{mob} for which $f(s)$ becomes true while the query is active. Given a query duration $t = \min(t_{\text{reply}}, t_{\max})$, where t_{reply} denotes the time at which a sensor has reported having found the object o, this effort q_{mob} can be modeled as

$$q_{\text{mob}} \sim A^{\circ} m t \tag{2}$$

where A° denotes the circumference of the scope area A and m denotes a factor representing the mobility of users. Therefore, both query success and communication effort are expected to rise with t and with m.

After a query installation at an object sensor s, object sensing will be performed continuously until $t_{\max}$ expires. The mobile device associated with s un-installs the query autonomously either when $f(s)$ becomes false or $t_{\max}$ is reached.

A query is declared successful if some object sensor s reports having found o at time t_{reply} with $t_{\mathrm{reply}} \leq t_{\mathrm{max}}$. The current position of s represents the location at which the object was found and will be included in the reply issued to the user (if a user-defined name is associated to the location of s, a reply will look as in Figure 1(d)). Once the first report is received, it could be useful to uninstall the query at all participating sensors (sending an additional $q_{\mathrm{init}} + q_{\mathrm{mob}}$ messages) in order to avoid useless "object-found" reports if sensors come in range of the object at a later time. However, as the number of such useless reports is usually much smaller than $q_{\mathrm{init}} + q_{\mathrm{mob}}$, we do not explicitly uninstall the query, but instead let each sensor s autonomously remove the query on the timeout t_{max} or when $f(s)$ turns false. Finally, a query is terminated without success, once the query timeout t_{max} is reached (in contrast, reaching q_{max} does not terminate the query – in this case, a reply may still be received by queried sensors due to user mobility).

In the remainder of the paper we will evaluate the presented query scoping service based on the search heuristics of Section 4. In particular, Sections 7 and 8 focus on evaluating the practicability of strategy I where we search an area close to the location where the object was last seen. Strategies II and III, similarly based on locations, depend on real-world data which is hard to model accurately. Nevertheless, one can use the considerations of Eq. (1-2) to relate the results we will present for strategy I to different search areas with different sizes. Finally, in the following Section 6, we evaluate the simple heuristics IV by means of a small user study performed in our office environment – the obtained results will then serve as a validation of the results obtained through simulation.

6 Real-world Experiment

The crucial question is whether the object search system can perform well enough to be a useful application. For answering this question, we first come back to the scenario introduced in Figure 1, where the user is at home and tries to verify the whereabouts of a given object which was left at the office. The mobile phones of the user's officemates (e.g., Bob) are registered with the association service and thus are considered relevant object sensors.

Our experiment was performed with four users working on the same floor. The users were given mobile phones running the object search prototype already tasked to perform continuous object sensing for all objects (using repeated Bluetooth discovery) and reported them in regular intervals to the back-end database. Similarly, 10 BTnodes [6] representing tagged objects were distributed in various rooms of the same floor. Figure 3 shows the experiment's setup (tagged objects are shown as numbered circles while the offices of the four participating users are shaded).

Note that while Bluetooth may be too expensive and battery-intensive to be used as an object tagging technology in a future system, it nevertheless allows to test whether, given a future technology with similar radio range, the mobility of a few office colleagues suffices to detect a given object in reasonable time.

Each user's readings were reported to the database as *(user, time, obj_id)* tuples. We considered only readings obtained during core office hours (9 a.m.–5 p.m.) while all others were discarded, resulting in around 30 hours of data. Using the collected data, we could compute the reply time of a query for a given object o issued to the four

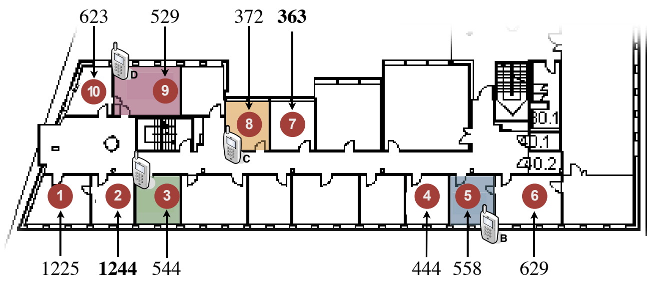


Fig. 3. Experiment setup

colleagues at an arbitrary point in time (say at time t_q): The reply time corresponds to the time between t_q and the next database entry on the queried object o.

Based on this consideration, we computed the expected average query reply time for each object, given that queries for this object were distributed uniformly over the experiment time. Note that non-office hours are simply "skipped" in this computation. In order to save messaging costs, user devices cached seen objects and only re-reported them to the database 10 minutes after their last report on the same object. This way, even if an object sensor has seen the object continuously, the resulting reports will yield an average query reply time of 5 minutes instead of zero.

For each object, Figure 3 shows the average reply time in seconds. Intuitively, we obtain better results for objects with a participating user in the same room. Further, note that the best results were obtained for objects close to the printer and the coffee machine (objects 7 and 8), while the worst results are for objects in rooms that were not visited by the participants during the time of the experiment.

We show a cumulative density curve of the observed reply times for object 2 (with worst results), object 7 (with best results), and the average over all objects in Figure 6(a). In all cases, reasonable success rates can be obtained with a maximum query time $t_{\max}$ of 30 minutes.

7 Model

In the last section, we focused on a small and confined search area and query scope. In the remaining sections, we use simulations to investigate the characteristics of an object search system operating in the wide-area with a larger user base, which provides us with a basis for the design of such a system given certain environmental conditions.

Note that adequate models of a future execution environment are hard to obtain, as these must consider many aspects of daily life. To provide an accurate basis for the design of an efficient system, models must define the number of participating users, the frequency at which these users lose or search for certain objects, particular scenarios in which objects are lost, the average number of tagged objects owned by each user, and so forth. Intuitively, such a model contains many parameters which cannot be influenced by the system designer. We call these *environmental parameters*. Our approach to these parameters is to investigate a significant portion of the (unfortunately) vast parameter space.

On the other hand, there are many *design parameters* which determine the system's performance and can more or less directly be set and varied by the system developer. These include the size of the search scope (the number of users that participate in searching for an object), the sensing range of an object sensor (which can be influenced by employing more expensive tag and object sensing hardware), or the timeout used for queries. For these *design parameters* we aim to find the most appropriate values, i.e., the parameter settings which can implement object search with the least communication overhead.

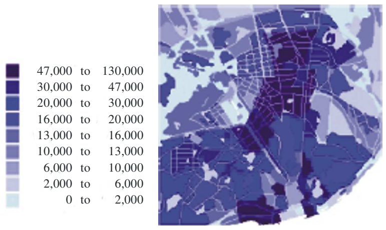
Scenario and Metrics. In the evaluated scenario, a user misplaces an object o and later issues a search query to the global query service. We assume that at the time the object left the range of the integrated object sensor, the user's mobile device recorded its location p. This location p will act as a hint for the search (implementing the presented heuristics I). We will evaluate two versions, a *cell-based* version in which p is a cell, and a *position-based* version in which p is an actual geographic point measured with a certain positioning error. In both versions, query scoping is performed according to Section 5.

The main metric we observe is the *success rate* of our system. This rate corresponds to the fraction of queries upon which a notification from some object sensor is received within the query timeout $t_{\max}$. Further, we will examine the overhead for query distribution q_{total} and the contained part q_{mob} which is caused by user mobility.

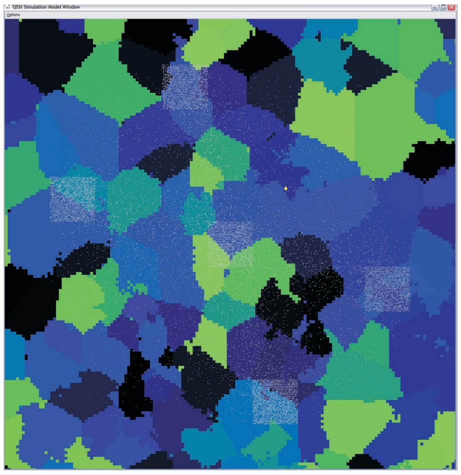
In our experiments, we will not examine object sensing costs explicitly, as we expect wide-area query installation to dominate the total cost due to the object sensor's much shorter wireless range and potential energy efficient optimizations of the object sensing implementation (e.g., object sensing could be performed only after a user has moved).

Environment Model. We assume that the object is left in a densely populated urban environment. In this setting, we will study how an object can be found by users who move according to pedestrian mobility models (see details on the mobility models below) in a square area of 1 km^2. The choice of the user density u_d is derived from the total population during the day in an urban area estimated in the Momentum project [17] (a downtown Lisbon example which we cite from [17] is shown in Figure 4(a)). For urban environments, the authors estimate the fraction of users with "pedestrian" mobility patterns as around 50%-70%[17, p. 37] from the total. The total includes other users who are assumed to be stationary or moving differently, e.g., with higher speeds on streets; we omit these users in our simulations. Moreover, as we are only interested in users associated with *a single* mobile provider, we chose more pessimistic values for the user density u_d: We will vary u_d from 100-2000 users/km^2, values which represent around a hundredth to a tenth of the estimated total daytime population. The default value for u_d is 500 users/km^2.

As mentioned, in some settings we use cell identifiers for positioning. To study such scenarios, we used actual position and orientation data from UMTS antennas together with a detailed model of land use types (e.g., buildings, highways, open, water) provided by the same project to compute the identifier of the strongest-signal cell for each point of the simulation area [17, 18]. In Figure 4(b), we give an example of a resulting cell-coverage map computed for a UMTS network of downtown Lisbon. For cell-based scenarios the simulation area is enlarged to 10 km^2 to avoid border effects with fairly large cells. We are aware that in reality several cells may be observed at a given location at different points in time, depending on dynamic factors such as interference, fading,



(a) Lisbon population distribution per km^2 during the day [17]



(b) Lisbon cell coverage and campus model

Fig. 4. Environment models

or user mobility. Because of this effect, in the *cell-based* scenario additional cells would need to be searched to be successful.

In this study, however, we assume that the object can be found in the cell where it was last seen. Results for scenarios in which several cells need to be searched to cover a certain location could be extrapolated from the results we provide.

Mobility Models. Generally, the tagged object will not move once its owner left it somewhere. In turn, the users' mobility model is a crucial aspect in our evaluation, as it determines the coverage obtained by object sensors carried by users. Therefore, the user densities we assumed in our simulations are quite small compared to actual densities observed in urban environments, on campus, or on office floors. In this regard, the fraction of simulated users represents the subset of all users who move according to the model we simulate. Additional users, e.g., sitting in their offices or moving differently, may then only improve results.

In the most basic setting, we use a random waypoint mobility model parameterized for pedestrian users. Users pick a random destination and start moving towards it with a speed drawn uniformly from (2,4) km/h. (The average speed of 3 km/h is chosen according to the ETSI guidelines [19].) As our simulation area can be fairly large, a trip's destination is chosen to be within 200 m from the user's current position.

We will also use a second mobility model which was derived from user WLAN traces observed on the Dartmouth campus [20]. The model includes *hotspot* regions which represent central points of the campus (for example, a hotel, a library, or a cafeteria). These hotspots tend to contain many users and also represent popular destinations chosen by the campus population. In our adaptation, we use five hotspot regions out of which one is in the middle and the other four shifted to each side of our simulation area. Each hotspot region's size is one hundredth of the simulation area. Half of the trips of a given user are made inside the current hotspot and half are directed to another (arbitrary uniformly drawn) hotspot on the campus. The remaining (non-hotspot) area is called the *cold* region. In our implementation, users never choose a destination in the cold region but only travel through it. As hotspot regions have a higher density, their positions and sizes are apparent in Figure 4(b), which shows a total density of 1000 users/km^2 in a 10 km^2 area of downtown Lisbon.

As in [20], the chosen trips include (in our version 2 to 5) waypoints, which are uniformly drawn from the rectangle between the user's source and destination and visited in the order of their distance to the source. The chosen speed and pause times follow log-normal distributions parameterized according to [20, table 3]. Note that the pause time distribution has a mean of 0.71 hours with a high standard deviation of several hours as the original paper found that users tend to stay in a hotspots for longer periods.

To avoid an initial transient period, we used initializations of user trips according to the perfect simulation method [21]. In the campus mobility model, however, some distributions had to be estimated, and thus a transient period of 1000 s remains. Object search queries are issued after this period.

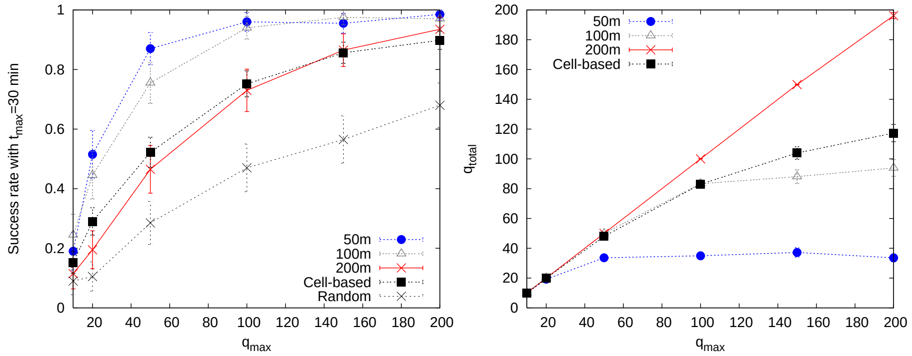
Sensor Model. A mobile device operates its object sensor continuously as long as a query is installed and running. In the default case, we assume that the object sensor has a sensing range of 5 m, that is, the object sensor sends a notification once the user carrying it comes within 5 m of the sought object.

Moreover, in some simulations we assume that the user has a position sensor available (e.g, GPS). To model the sensor's localization error, the position returned by the sensor is drawn uniformly from a disk centered around the actual position of the user. We refer to the radius of this disk as the *positioning error* e_p used in the simulation (e_p is set to 100 m if nothing else is stated). Note that with this error distribution, the density of observing an actual error, say e, is proportional to the circumference of a circle with radius e, and therefore the mean error is $(1/\sqrt{2})e_p$. This positioning error occurs not only when the owner's mobile device records the position p where it has last seen an object, but also when distributing a query to object sensors near p, as these object sensors' positions are measured with an independent positioning error.

Alternatively, we also model a scenario in which the positioning sensor simply returns the identifier of the UMTS cell to which the mobile phone is currently connected. This is a worst case scenario as most cell-based localization approaches combine signal strength information from multiple nearby cells together with antenna positions to obtain more accurate localization results.

Scoping. As mentioned, in the position-based version the scope will consist of a disk with a certain radius r around the position hint p. Because at the time when p was measured the object is out of range of the user's object sensor, we set r as $r = s_r + e_p$ where s_r denotes the range of the object sensor and e_p the positioning error. Similarly, if p is a cell id, the scope will consist of the object sensors served by the cell p.

Note that in this model, the object really lies within the computed scope. While this is not always true in reality, our simulation focuses on evaluating the system's performance for situations in which the search strategy is in fact correct. If in reality, the search strategy should fail in q percent of all cases, the resulting success rates can be extrapolated from the results we provide.



(a) Success rate

(b) Sent messages

Fig. 5. Success rate and overhead with different positioning technologies

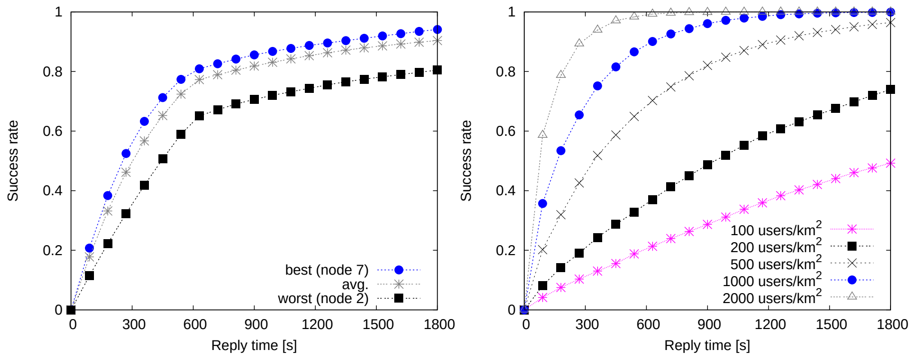
8 Evaluation

Using the simple scenario and the environment models described above, we aim to investigate several aspects of a future object sensing system. Foremost, given some scope, we want to confirm whether it is possible to find objects with reasonable success rates and small-enough overhead. Further, we aim to investigate how large cell-based scopes compare to position-based scopes and to a random query dissemination strategy which queries a certain fraction of all users. Moreover, we aim to gain insights into the sensitivity of the system's performance with regard to parameters such as user mobility, object sensing range, or chosen query timeouts.

8.1 Success rate

In the first set of simulation runs, we investigated the query success rate observed with *position-based* scoping and *cell-based* scoping. Figure 5(a) shows the fraction of successful queries (upon which the queried sensors have located the object within 30 minutes), when the user-imposed limit $q_{\max}$ denoting the maximum number of queries is varied. Five different graphs show the results obtained with different positioning errors e_p (from e_p=50 m to e_p=200 m), cell-based scoping, and a random strategy where we distribute the query to a fraction of $q_{\max}/500$ of all users. For all graphs, the obtained success rate can be increased by raising the maximum number of queries $q_{\max}$ and reaches acceptable levels of above 90% with $q_{\max}$=200.

The number of messages q_{total} which were actually sent in the same runs is shown in Figure 5(b). As, by the definition of our protocol, the search area becomes larger with an increased positioning error, the required effort increases as well. Similarly, searching the coverage area of the whole cell where the object was left requires sending more messages before obtaining reasonable success rates. Note, however, how in Figure 5(b) the actual number of sent queries q_{total} at some point stops growing with the user-imposed limit of maximum queries $q_{\max}$. This is because with small enough scopes the object is found before the maximum message limit $q_{\max}$ is reached. Observe also,



(a) Real-world measurements (b) Random waypoint simulations

Fig. 6. Cumulative density functions of reply times

how the performance of cell-based scoping is comparable to a 200 m positioning error and even outperforms the latter in terms of q_{total}. This is because with position-based scoping and large positioning errors, many ineffective queries are sent to mobile devices which erroneously measured a position which was close to the position hint p.

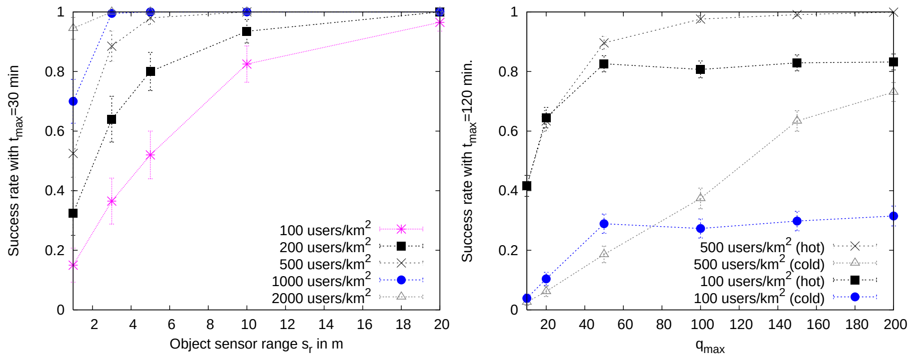
Finally, as Figure 5(a) shows, any scoping performs better than a random strategy. Even if 40% of all users are queried, the success rate is still only around 60%. Needless to say, the communication effort of the random strategy is worst as it is proportional to the total number of users (not shown).

8.2 Timeout, sensing range, and different mobility models

Apart from scoping, several other parameters may significantly influence the performance of the system.

The first such parameter is the timeout used for queries. Here, it is not clear, *when* the replies were received and whether a more adequate choice of the timeout (previously set to $t_{\max}$=30 min) can be made. Note that choosing an adequate timeout is particularly relevant when *object sensing itself* is considered a significant cost. Especially because in reality the object might be outside the chosen scope, it is important not to sense in vain for too long, but at the same time to issue a confident "not-found" reply. Further, the system performance is expected to vary with the user density. We show the interplay of these two parameters with position-based scoping in Figure 6(b). Each graph represents the cumulative density function of the reply time obtained after 5000 repeated simulation runs (each data point represents the fraction of requests answered within the given timeout) if no message limit $q_{\max}$ is imposed. As expected, the likeliness of finding the object increases with a longer timeout, but for high user-densities very short timeouts are already sufficient. Moreover, very good success rates can be obtained with $t_{\max}$=30 min, even with user densities down to 500 users/km^2. For lower densities longer timeouts must be used.

Observe that the graphs of Figure 6(a) measured in our office floor experiments, in which the actual user density was greater than 4000 users/km^2, are comparable with



(a) Sensing range (b) Campus mobility

Fig. 7. Varying sensing range and mobility

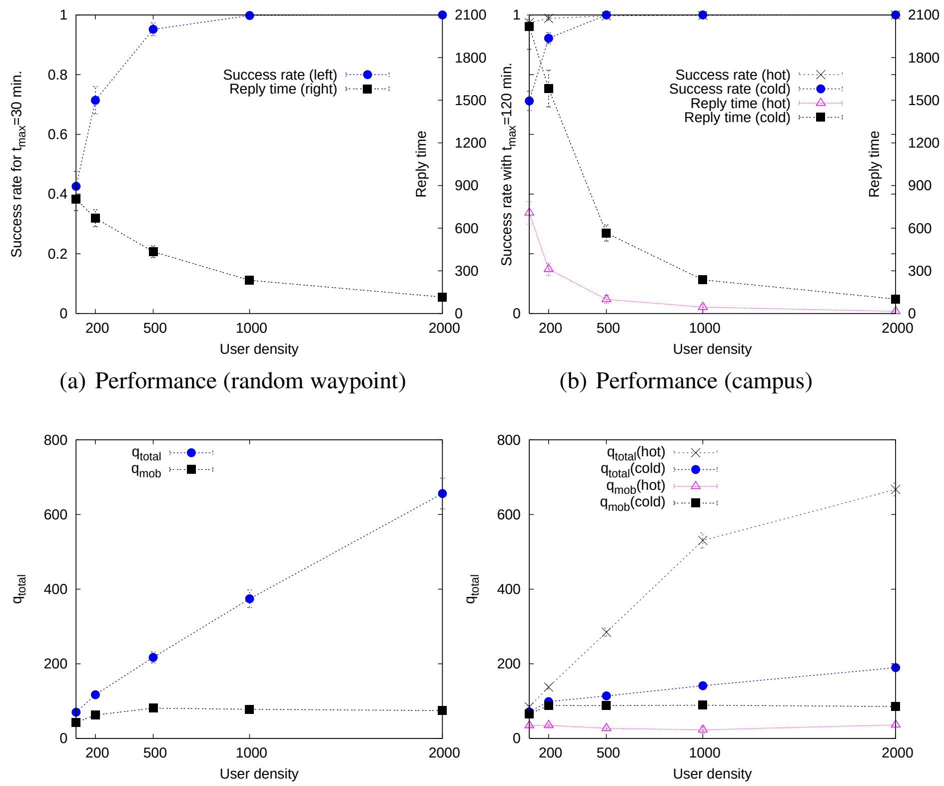
user densities of 500 to 200 users/km^2 in Figure 6(b). This is compatible with our earlier conjecture that the random waypoint simulation only models the "pedestrian" fraction out of the total users, and confirms that the approach to look at user densities which are smaller than in reality is valid.

A second important parameter, which is expected to have a large impact on the performance, is the sensing range of the employed object sensors. In the runs shown in Figure 7(a), we demonstrate the impact of the sensing range on the success rate of position-based scoping. With 2000 users/km^2, even a sensing range of 1 m yields acceptable results. As expected, however, the sensing range has a high impact. When designing a future system that shall be robust to small user densities, it seems worthwhile to invest in object sensing technology with a higher range.

Finally, a third crucial parameter is the mobility of the system's participants. Here it is unclear whether the random waypoint model used is perhaps too optimistic. To analyze this, the simulation results of Figure 7(b) show the success rate with the campus mobility model when raising the message limit $q_{\max}$. We show four graphs for the cases in which the object was left in a hotspot or in the cold region with two different user densities. Because pause times in this model are quite long, we extended the query timeout $t_{\max}$ to 2 hours. Note, however, that the total number of queries remains limited to $q_{\max}$ and therefore the results remain comparable to earlier simulation runs shown in Figure 5(a). Here, for very small user densities, the success rate cannot be improved by raising $q_{\max}$ as the timeout remains the dominating constraint. For 500 users/km^2, however, the object can often be found with at most 200 messages even if it lies in the cold region.

8.3 Effects of increasing user density

Additional lessons can be learned when observing our metrics' sensitivity to an increasing user density. These experiments were performed with cell-based scoping and are shown in Figure 8. We show the success rate and the query reply time while varying the user density in Figure 8(a), and analogously the results for the campus mobility



(a) Performance (random waypoint)

(b) Performance (campus)

(c) Overhead (random waypoint)

(d) Overhead (campus)

Fig. 8. Cell-based scoping when the user density is varied

model in Figure 8(b). The corresponding overheads are shown in Figure 8(c) and 8(d), respectively. Note that for these runs no limit $q_{\max}$ is set.

Both overhead figures include the total overhead q_{total} and the overhead due to user mobility q_{mob} included in the total. It is interesting to observe that q_{mob} does not increase with higher user densities. We explain this by the fact that the query reply time decreases with increased user density and therefore compensates for the expected increase in the mobility-based overhead. Quite differently, q_{init} (equal to $q_{\text{total}} - q_{\text{mob}}$) increases proportionally to the user density as the number of queries is not limited by a certain $q_{\max}$.

The main result here is that once the success rate is good, an increased number of messages is "wasted" towards lowering the reply time. In other words: waiting for users to move is more efficient than simply querying more users. As a consequence, if a higher reply time were acceptable, then the protocol can do with much less queries (e.g., by computing q_{init} as if the density were 500 users/km^2).

Summary. Summing up, we observed that high success rates can be obtained with a range of different mobility patterns and scoping variants. Cell-based scoping, which is free from additional overhead in propagating object-sensor position information, proved to be particularly valuable. Finally, in certain circumstances the system may even work with very low user densities which represent a hundredth of the expected daytime population in an urban area.

9 Conclusion

In this paper we presented the architecture, design, and evaluation of an object search system relying on mobile phones as omnipresent object-sensing devices. Based on the ubiquitous mobile network infrastructure which is already in place, wide-area search for everyday objects becomes possible without incurring the high costs involved in instrumenting a larger environment with an object-sensing infrastructure.

Our system makes use of an unconventional approach, which relies on the participants' mobility in order to cover an essential portion of the users' space. We therefore spent significant effort on modeling and testing the circumstances in which such an object search system would be used in the large. The results are encouraging. In all our experiments, we could observe a high rate of successful queries, that is, of objects being found. While the time until a reply can be obtained varies with user mobility and density, our conjecture – that most of the time an object found event will be received eventually – was confirmed. Moreover, we could show that even in settings with high positioning errors or which rely solely on the observed cell id for localization, the total overhead for distributing an object search query remains acceptably low. While this does not change the basic fact that objects left in deserted places will not be found, we showed that for objects left within the users' space such a system is feasible.

In a broader context, this paper has analyzed the properties of the coverage obtained from user-carried sensors. By means of the average query reply time we observed with a certain sensing range, participant density, and mobility pattern (e.g., 30 minutes), we have quantified the time which must pass before a point-shaped phenomenon has been sufficiently covered. Therefore, the query reply time can be interpreted as the (reciprocal of the) maximum sampling frequency that our user-centric infrastructure can implement for a certain spot of the area under observation. If the phenomenon changes only insignificantly between samples, then coverage is sufficient. For example, recent work has mentioned measuring air quality or average noise levels [22] in urban areas. In such systems, the examined trade-offs between sensing range, maximum sampling frequency, and participant density are likely to re-appear.

In future work, we aim to collect real-user data using our prototype implementation (e.g., data on the participants' social networks) and use it to test the effectiveness of user-profile based heuristics which were not evaluated so far. Further, current work focuses on extending our model with an aggregated cost measure which integrates the costs of query dissemination with the costs of object sensing.

Acknowledgments. We would like to thank Michael Fahrmair, Friedemann Mattern, and Daisuke Ochi as well as our anonymous reviewers for their thoughtful comments and suggestions on draft versions of this paper, and Chie Noda for contributing to the presented work in earlier phases of our research collaboration. Moreover, we would like to thank Hans-Florian Geerdes for valuable advice on the Momentum dataset [18] and Christian Floerkemeier for pointing out suitable hardware options for the implementation of a future object search system.

Bibliography

[1] Want, R., Fishkin, K., Gujar, A., Harrison, B.: Bridging Physical and Virtual Worlds with Electronic Tags. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'99), Pittsburgh, PA, USA (1999) 370–377

[2] Decker, C., Kubach, U., Beigl, M.: Revealing the retail black box by interaction sensing. In: Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS'03), Providence, RI, USA (2003)

[3] Yap, K.K., Srinivasan, V., Motani, M.: MAX: Human-centric search of the physical world. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SENSYS'05), San Diego, CA, USA (2005)

[4] RF Code, Inc.: Mantis^{TM} active RFID tags 433 MHz data sheet. `www.rfcode.com/data_sheets/433_mantis_tags.pdf` (2006)

[5] Wibree Technology. `www.wibree.com` (2006)

[6] BTnodes. `www.btnode.ethz.ch` (2006)

[7] Borriello, G., Brunette, W., Hall, M., Hartung, C., Tangney, C.: Reminding about tagged objects using passive RFIDs. In: Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp'04), Nottingham, England (2004)

[8] Shimizu, H., Hanzawa, O., Kanehana, K., Saito, H., Thepvilojanapong, N., Sezaki, K., Tobe, Y.: Association management between everyday objects and personal devices for passengers in urban areas. Pervasive 2005, Demonstration, Munich, Germany (2005)

[9] Gibbons, P.B., Karp, B., Ke, Y., Nath, S., Seshan, S.: IrisNet: An architecture for a worldwide sensor web. IEEE Pervasive Computing **2**(4) (2003)

[10] Pietzuch, P., Ledlie, J., Shneidman, J., Roussopoulos, M., Welsh, M., Seltzer, M.: Network-aware operator placement for stream-processing systems. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), Atlanta, GA, USA (2006)

[11] Engberg, S.J., Harning, M.B., Jensen, C.D.: Zero-knowledge device authentication: Privacy & security enhanced RFID preserving business value and consumer convenience. In: Proceedings of the 2nd Annual Conference on Privacy, Security and Trust (PST'04). (2004)

[12] Trossen, D., Pavel, D.: Building a ubiquitous platform for remote sensing using smartphones. In: Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MobiQuitous'05). (2005) 485–489

[13] Bolliger, P., Langheinrich, M.: Distributed persistence for limited devices. System Support for Ubiquitous Computing Workshop at Ubicomp 2006 (Ubisys 2006) (2006)

[14] Laasonen, K., Raento, M., Toivonen., H.: Adaptive on-device location recognition. In: Proceedings of the 2nd International Conference on Pervasive Computing (Pervasive'04), Vienna, Austria (2004)

[15] Smith, I., Consolvo, S., LaMarca, A., Hightower, J., Scott, J., Sohn, T., Hughes, J., Iachello, G., Abowd., G.D.: Social disclosure of place: From location technology to communication practices. In: Pervasive 2005, Munich, Germany (2005)

[16] Frank, C., Roduner, C., Noda, C., Kellerer, W.: Query scoping for the Sensor Internet. In: Proceedings of the IEEE International Conference on Pervasive Services (ICPS'04), Lyon, France (2006)

[17] Ferreira, L., Correia, L.M., Xavier, D., Vasconcelos, A., Fledderus, E.: Deliverable d1.4: Final report on traffic estimation and services characterisation. Technical Report IST-2000-28088, Momentum Project (2003)

[18] Momentum: Models and simulation for network planning and control of UMTS. `momentum.zib.de/data.php` (2006)

[19] ETSI: Selection procedures for the choice of radio transmission technologies of the UMTS. Technical Report 3.2.0, European Telecommunications Standards Institute (1998)

[20] Kim, M., Kotz, D., Kim, S.: Extracting a mobility model from real user traces. In: Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'06), Barcelona, Spain (2006)

[21] Le Boudec, J.Y., Vojnović, M.: Perfect simulation and stationarity of a class of mobility models. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05), Miami, USA (2005)

[22] Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., Srivastava, M.B.: Participatory sensing. In: Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications, Boulder, Colorado, USA (2006)