# A Sensor Network Abstraction for Flexible Public Sensing Systems

Damian Philipp, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
Email: [damian.philipp,frank.duerr,kurt.rothermel]@ipvs.uni-stuttgart.de

***Abstract*—Public Sensing is a new paradigm for developing large-scale sensor networks at low cost by utilizing mobile phones that are already surrounding us in our everyday lives. In this paper we present a sensor network abstraction layer for creating flexible public sensing systems that can execute arbitrary queries. To this effect we develop several algorithms to select mobile nodes for executing a query. These algorithms allow a user to define a trade-off between quality and efficiency of query execution by choosing an appropriate algorithm. Our evaluations show that we can achieve a 99% increase in efficiency with the most efficient approaches and only about 10% decrease in result quality under worst conditions.**

## I. Introduction

Public Sensing (PS) is a new paradigm for developing large-scale sensor networks at low cost by utilizing mobile phones that are already surrounding us in our everyday lives [1], [2]. Liu et. al. [3] created a theoretical basis for PS by showing that the area covered by a sensor network is significantly increased when the sensor nodes are mobile. In contrast to previous approaches, where the movement of nodes is controlled by the system, PS exploits the inherent uncontrolled mobility of people carrying smartphones.

From a technical perspective, PS is leveraged by the proliferation of commodity smartphones like the Apple iPhone or the Google Android phones. These devices offer features that were previously found only in expensive specialized sensing devices, e.g., Tmote Sky [4]. A WiFi interface allows for fast short-range communication between nodes, a UMTS interface allows for internet access at any time. Each node can determine its location through systems such as GPS, WiFi-Positioning [5] or ultrasonic positioning systems [6]. Smartphones also contain environmental sensors – at the very least for measuring light intensity (camera) and volume (microphone). Other sensors can easily be added via Bluetooth.

In this work we present a novel kind of opportunistic public sensing system (PSS) [7] that allows for flexible and efficient acquisition of arbitrary sensor readings in arbitrarily large areas, limited only by the availability of nodes carrying the necessary sensors. Our system can handle both one-shot and periodic queries as well as the acquisition of several types of sensor readings in parallel. Up to now, research has mostly focused on either creating large-scale maps of a single kind of data [8], [9] or on arbitrary sensing tasks in a small area [10]. We face several challenges in creating a large-scale PSS.

First, an application interface is required that allows for posting queries to the PSS in an intuitive and flexible manner.

Consider the example of monitoring environmental variables, e.g., noise levels. Excessive noise is considered to be an environmental hazard [11]. It may, therefore, be of interest to people to create maps of noise levels on the streets of their hometown or their work environment. As noise levels may vary over a given area, it is important to achieve a good spatial coverage in the area of interest to get a meaningful result. However, determining which level of coverage is "good" highly depends on the application.

From this example, we can identify the following requirements: The user must be able to explicitly specify the points of interest defining the service area of a query, since for example using simple grid-based approaches to guarantee a certain coverage may result in readings taken at indoor locations, which are of no use in our example. This specification must be independent from the mobility of devices, as the user cannot be expected to have intimate knowledge on the current state of the PSS, e.g., where mobile nodes are located. In addition, the user must be able to specify how many readings shoud be taken at each point of interest (POI). One reading per POI might not be enough, e.g., when a user wants to use an aggregation function to remedy for outliers due to faulty sensor equipment.

Our first contribution is, therefore, an intuitive sensor network abstraction of a PSS to be presented to the user. Users give a mobility-transparent description of points that are relevant to them as the design for a static (wireless) sensor network comprised of virtual sensors. This specification includes the number of readings to take at each sensing site, thus allowing for $k$-coverage of sensing sites. Our abstraction layer will select suitable mobile nodes from the PSS to fill the role of each virtual sensor. Contrary to related work [12]–[14], our system does not need a training period but will instead start reporting sensor readings instantly.

The second challenge in creating a large-scale PSS is selecting the nodes to provide readings for virtual sensors. The first issue is the mobility of nodes. Consider the noise monitoring application from the previous example. Sound levels change when the distance between the source of the sound and the sensor changes. To obtain accurate readings, it is, therefore, necessary to chose nodes close to each sensing

Published in 8th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS'11), pages 460-469, Valencia, Spain, 2011.
© IEEE 2011
http://dx.doi.org/10.1109/MASS.2011.52

site. Due to the uncontrolled mobility of nodes, we can not know in advance which node will be close to which site. Therefore, we developed algorithms for on-demand selection of nodes to obtain sensor readings.

Determining which node should take a reading for a given point is further complicated by uncertain position information. For instance, GPS accuracy is rarely better than 5 meters [15], while less energy-intensive approaches like WiFi-based or cell-based positioning are even less accurate [5].

Finally we need to consider the energy consumption of our system. In traditional sensor networks, energy is a severely limited resource. While a smartphone has a rechargeable battery, energy is still a severe limitation. As the public sensing application will run as a continuous background task on every node, we must take care to limit energy usage so that the battery lifetime of the phones is not decreased significantly. Intuitively speaking, even with our system running, the battery of a phone should not run out before the end of the day.

To quantify the above issues, we identified several metrics for the performance of our system as a second contribution. These metrics can be divided into two classes. Quality metrics, i.e., coverage of observation area and the distance of readings to requested sensing sites, measure the quality of the query result. Efficiency metrics, i.e., the amount of redundant readings obtained, the sensing load of each individual node, and the network load, measure the cost of the sensor data acquisition process. We also give a short discussion on energy cost for communication. While the overall energy consumption would be a more powerful efficiency metric, we cannot evaluate it, since there is no comprehensive energy model available for the various types of sensors that can be used by our system.

These metrics form the basis for the third contribution: Distributed algorithms for selecting mobile nodes to obtain sensor readings. The goal of these algorithms is to ensure a certain result quality while causing minimum cost as measured by our efficiency metrics.

The remainder of this work is structured as follows: Section II presents related work. Section III will introduce our system model, followed by the definition of our sensor network abstraction in Section IV. We present our quality metrics in Section V. The algorithms for sensor selection we developed are presented in Section VI and subsequently evaluated in Section VII, along with a presentation of our efficiency metrics. We will conclude with a summary in Section VIII.

## II. Related Work

PS has generated a lot of interest in the research community in recent years [1], [2]. Several prototype systems that have mobile nodes carrying their own sensors, such as AnonySense [16], CenceMe [17], MobGeoSen [18], Bubble-Sensing [10], CO-monitoring [8] and noise pollution monitoring [9] have been developed. Each of these systems is built for a single task only, e.g., measuring air quality or sensing social context, whereas our system allows for performing several of these tasks in parallel. They all have in common that participating nodes constantly sample their sensors and upload the samples to a central server. No coordination or optimization is performed. MetroSense [19] is a general framework that discusses several high-level topics and optimizations for PS but does not present actual solutions to algorithmic challenges.

In our previous work on PSS, we considered a system where mobile nodes carry communication equipment only [20]. In this system RFID sensors are located at fixed positions and can be sampled by the mobile nodes. The nodes coordinate so that sensors are read only once per measurement interval. Unlike our current work, the queries that can be executed in this system are limited by the type and number of RFID sensors that are deployed. In addition, as the position of each RFID sensor is known exactly, the distance of a node to the sensing site does not impact the accuracy of the result.

We also presented another system for obtaining measurements along street segments which uses mobility prediction and coordination between nodes to achieve $k$-coverage [21]. This system takes continuous readings as opposed to the point-wise readings in our current work.

MapCorrect [22] presents a system for automatic validation and correction of road maps from GPS fixes. A gateway uses a pessimistic algorithm to estimate when nodes should take GPS fixes, thus reducing the number of redundant sensor readings.

Reddy et. al. [12]–[14] developed a reputation system for selecting the most suitable nodes for long-running tasks. The reputation is built over a long time (e.g., several days) from geographical and temporal availability, derived from mobility information and from past participation in sensing tasks. However, the purpose of the system is to aid a human operator in node selection. No automatic node selection is performed.

In the context of traditional sensor networks, reducing the number of sensor readings that are taken has been a topic of interest for quite some time. There are two main approaches: coverage optimization and model-driven optimization. In coverage optimization approaches sensors are assigned a covered area and sensors are chosen so that the areas of chosen sensors have the least overlap [23], [24]. Model-driven approaches [25]–[28] determine the information value of sensors and choose only those sensors with the highest information value. None of these systems have the problem of determining what node to query for a sensing site, as sensing sites are predetermined by the sensor deployment.

Works in actuated sensing [29], [30] also focus on networks of mobile nodes. The main difference to PSS is that the movement of nodes is controlled by the system. Thus, these systems achieve the benefit of using fewer sensors to cover a large area at the additional cost of having to deploy specialized mobile sensor nodes.

## III. System Model

Next, we present our system components and assumptions. Our system consists of a gateway and a number of mobile nodes. The gateway is located somewhere on the internet and serves as an interface for users and applications to submit queries to the PSS. It is responsible for distributing queries in the system and serves as a sink for sensor readings.

Each node consists of a WiFi interface, a UMTS interface for mobile internet access, a built-in GPS device and a set of environmental sensors (light, sound, temperature, air pollution, etc.). We assume that the GPS sensor does not provide perfect location information. The error for position fixes follows a two-dimensional normal distribution with zero-mean and standard deviation $\sigma$. For each position fix, the GPS sensor also returns the current $\sigma$.

Let us now define some terminology. $\delta(a,b)$ is used to denote the Euclidean distance of points $a,b$ in the 2D plane. We will use $pos_{true}(r)$ and $pos_{vis}(r)$ to refer to the true position and visible position of a reading $r$ respectively. These positions are not necessarily identical, as the system cannot always determine $pos_{true}(r)$ precisely, e.g., due to uncertain positioning information or due to node movement in the time between taking a position fix and taking a sensor reading. We extend this notation to all other kinds of positions as well.

The variables monitored by the environmental sensors follow an unknown spatial distribution. Although the concrete distribution is unknown, we assume that two readings $a$ and $b$ taken some distance $\delta(pos_{true}(a), pos_{true}(b))$ apart, their difference is limited by an (unknown) monotonically increasing error function of the distance: $|a - b| \leq e(\delta(pos_{true}(a), pos_{true}(b)))$. Readings from an environmental sensor are returned instantly upon accessing the sensor.

Nodes use their WiFi interface to send one-hop broadcasts to their neighbors. We assume every node to have continuous internet access through multi-hop routing in clusters of mobile nodes. In such a cluster, only the cluster head (CH) uses its UMTS interface for internet access. Messages from the gateway are sent to the CH and then relayed via WiFi broadcast, messages from cluster members other than the CH are collected at their respective CH and then sent via UMTS. Intra-cluster routing is done using a standard ring-based routing approach. Cluster membership of a node is determined by the hop count from the node to a CH. Nodes always joins the cluster of the closest CH, if that CH is closer than some system defined maximum hop count. If no CH is available, unclustered nodes elect a new CH, thus forming a new cluster.

For the following considerations we assume that our system will be deployed in an urban setting. Nodes move at walking speed. We also assume that every node has several other nodes in its one-hop WiFi neighborhood, although our system operates independently of the actual state of the WiFi neighborhood.

## IV. Sensor Network Abstraction

The base of our sensor network abstraction is the concept of a virtual sensor. A virtual sensor has several properties: a position, called the sensing site $s$, a type $t$ of reading (light, temperature, noise, etc.) and a sensing period $p$, denoting a time interval at which readings will be taken.

To obtain data from the PSS, a user generates a set of virtual sensors $V$, thus defining the service area. She then submits a query $Q = (V, k, \delta_{max})$ to the gateway. Apart from the set of virtual sensors, the query contains two additional parameters

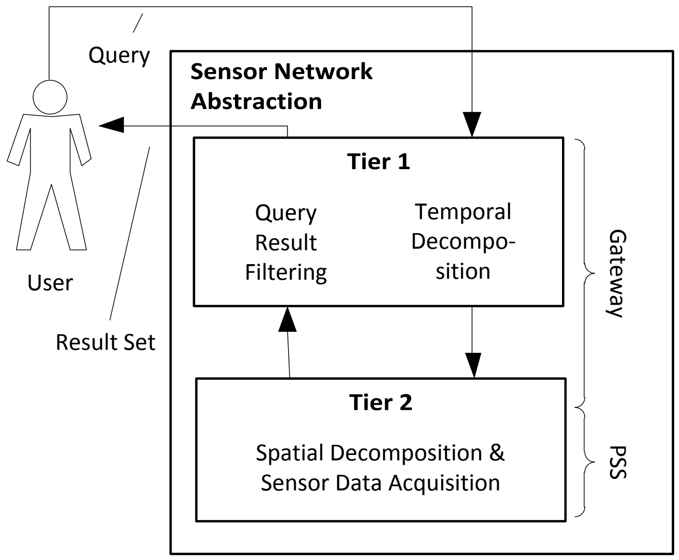

Figure 1. Overview of the system architecture

that have no resemblance in a static sensor network. $k$ defines how many readings should be obtained by the PSS for every virtual sensor. $\delta_{max}$ is used to define the covering relationship: A node $n$ is said to cover $s$, iff $\delta(n,s) \leq \delta_{max}$. For each sensing site $s$, our algorithms will obtain readings from nodes covering $s$. Thus, the selection of $\delta_{max}$ directly influences the maximum error of a reading by bounding the error function $e$. However, as $e$ is unknown to us, we push the selection of a suitable $\delta_{max}$ to the user, so that she can use expert knowledge to select a suitable value. Note that $V$ is unique for every query, thus virtual sensors are not shared across queries. Queries may be canceled by the user at any time.

To simplify the following descriptions, we assume that all virtual sensors of a query share the same values for $p$ and are of same type $t$. With this simplification, we can rewrite a query as $Q = (S, t, p, k, \delta_{max})$, where set $S$ denotes the sensing sites of the different virtual sensors. Modifying our algorithms for executing queries as in the original definition is straight-forward.

The sensor network abstraction is implemented as a two-tier architecture, depicted in Fig. 1. The first tier is the gateway which serves as a user interface to the PSS, performs filtering of query results and performs temporal decomposition of queries. The second tier is implemented by all mobile nodes participating in the system in a distributed way and is responsible for spatial decomposition of queries.

The PSS only supports the execution of one-shot queries, as we have to reconsider which nodes to select for taking readings at the end of each period due to the mobility of nodes. We thus perform temporal decomposition in the first tier, where the gateway issues a one-shot query to the PSS at the end of every period. Notice that our system does not place any restriction on the amount of queries that are submitted in parallel.

The second tier in our system is responsible for spatial decomposition of queries, that is, selecting nodes to fill the role of the virtual sensors, distribute the query to these nodes, and to transport their readings back to the gateway. Its implementation is split across the gateway and the mobile nodes as will become clear when we present the actual algorithms in Section VI.

## V. Quality Metrics

Intuitively speaking, a query result is of high quality if the data gives a sufficiently complete and accurate picture of the real world, i.e., no phenomenon was missed and measurements carry a bounded error. Whether or not given data can be considered complete and accurate is dependent on the application and thus out of the scope of the PSS. We have, however, identified two quality metrics that will allow a user to make this determination: coverage of queries and distance of readings.

Coverage of a query is defined as the fraction of sensing sites for which at least one reading was obtained. The environmental readings we are focusing on exhibit a spatial variance. Thus, a user can determine from the coverage value whether any phenomenon might have been missed or the data is sufficient. An increased coverage leads to an increased chance of capturing all properties of the observed variable and, therefore, to a more meaningful query result as well.

Note that in our definition of coverage we do not consider $k$-coverage. As it is impractical to integrate the number of readings taken at each covered site and our binary notion of coverage into a meaningful value, we inform the user about the number of readings taken at each site, so that he can again use expert knowledge to assign a quality value to these readings.

The distance of a single reading is defined as $\delta(pos_{true}(r), s)$, the distance of the true position $pos_{true}(r)$ where a reading was taken and the sensing site $s$ the reading is assigned to. As we assume that a reading may deviate from the true value at $s$ if $\delta(pos_{true}(r), s) > 0$, knowing $\delta(pos_{true}(r), s)$ indicates to the user the accuracy of the query result, i.e., how trustworthy the readings are. To determine the overall quality of a query result, the user is presented with the full set of reading distances and can then map these to a quality value according to his own requirements. However, as we know that the deviation is bounded by a monotonically increasing error function $e$, we can conclude that the quality of a query result increases as the single reading distances decrease.

## VI. Sensor Selection Algorithms

Based on the sensor abstraction and query definition, we can now present the algorithms for distributed query execution. The basic problem that we solve is that given a query $Q$, for each sensing site $s \in Q.S$, we want to return $Q.k$ readings $r_i$ with $i \in [1, Q.k]$ at a location $pos_{true}(r_i)$. To provide a high quality result, we should minimize the distance $\delta(s, pos_{true}(r_i))$ for each $s$ and maximize the overall coverage as well as matching the number of readings to $Q.k$. With perfectly accurate node positions, this goal is easy to achieve. However, we have to deal with position errors. Our algorithms execute queries instantly, thus we do not introduce any delay in waiting for node positions to improve with regard to our quality metrics.

Table I
Taxonomy of selection schemes

| | Independent | Coordinated |
|---|---|---|
| Probabilistic | Nearest-Neighbor Candidates | Coordinated Nearest-Neighbor Candidates |
| Deterministic | Deterministic Nearest-Neighbor<br>Naive | Coordinated Deterministic Nearest-Neighbor |

In addition, we also consider the efficiency of query execution. To this end, we design our algorithms for taking as few readings as possible, since every redundant reading increases the energy consumption on each node (reading the sensor, potentially processing the reading) and increases network load, as each reading has to be transmitted to the gateway.

All of our approaches follow the same outline:

1) Gateway sends $Q$ to relevant cluster heads via UMTS
2) Cluster heads relay $Q$ to all cluster members via WiFi
3) Nodes determine their location via GPS
4) Nodes execute selection scheme
5) Selected nodes take a sensor reading and send it back to the gateway via their cluster head
6) The gateway filters the set of returned readings and passes the result to the user

In Step 1, a cluster head is considered relevant if there is a possibility that at least one of its cluster members is covering any sensing site in $Q.S$. This can, for example, be evaluated by computing the potential coverage area (PCA) of a cluster. The PCA is a circle centered at the position of the CH with a radius of $(\text{WiFi comm. range} * \text{max. clustering hop count}) + Q.\delta_{max}$. If there is at least one sensing site in the PCA, the CH of that cluster is considered to be relevant.

To prepare for the sensor selection scheme, nodes determine their position $pos_{vis}(n)$ via GPS in Step 3. If the node later takes a reading $r$, we set $pos_{vis}(r) = pos_{vis}(n)$. In Step 4, every node that received $Q$ executes one of the selection schemes presented later in this section to determine whether or not it should take a reading. The implementation of this step is crucial to minimizing the number of redundant readings while at the same time ensuring the coverage of the sensing site.

The filtering operation on the gateway in Step 6 is quite simple. For each sensing site $s$, the gateway first computes the set of readings $R_s$ that were taken by nodes covering $s$. If $R_s$ contains more than $k$ readings, the reading $r'$ with the maximum reported distance $\delta(r'_{pos}, s)$ is removed from $R_s$. This is repeated until there are only $k$ readings left in $R_s$. Finally, for each sensing site $s$, $R_s$ is returned. To ensure that every reading is assigned to at most one sensing site, we require $\delta_{max}$ to be chosen accordingly: $\forall s, s' \in Q.S : \delta(s, s') > 2 * Q.\delta_{max}$

In the following subsection we present our different selection schemes. A taxonomy is depicted in Table I.

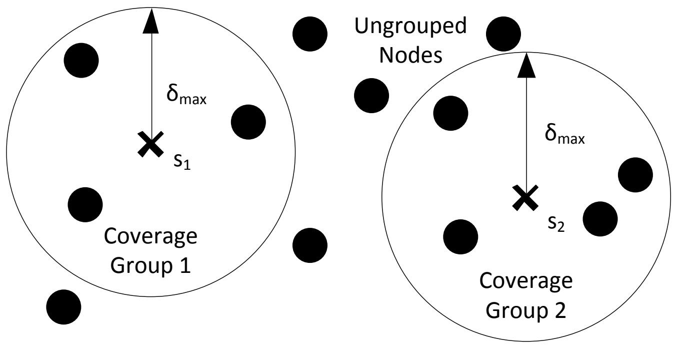

Figure 2. Nodes in the vicinity of sensing sites form a coverage group. Dots are nodes, crosses are sensing sites.

## *A. Independent Selection Schemes*

In a naive approach, the selection scheme is to select every node to obtain a sensor reading.The gateway can then choose readings from this set to achieve the best possible coverage of sensing sites and lowest distance of reading positions to sensing sites, thus yielding a high quality. The downside to this approach is that efficiency is very low: Every node has to take a reading for every query. We will see many redundant sensor readings in areas where there is more than one node close to a sensing site.

To improve the efficiency of query execution we present several optimized selection schemes. In this section independent selection schemes are presented, where nodes use only local knowledge to decide on whether they should take a sensor reading. Coordinated approaches, where nodes communicate to determine which nodes should take sensor readings, are presented in the next subsection.

We present two independent selection schemes where nodes first form coverage groups, determined by what sensing site they are covering (see Fig. 2), and then independently decide whether or not they will take a sensor reading. In all of our approaches, the nodes in one group have to be able to communicate directly. Therefore, we require that $\delta_{max}$ is chosen no larger than half the WiFi communication range.

The basic idea of the first algorithm, Nearest-Neighbor Candidates (NNC), is to probabilistically select nodes from a set of candidate nodes $s_{NNC}$ for each sensing site $s$. A node is included in $s_{NNC}$, if its distance to $s$ is at most $\delta_{limit} = \min(\delta_{max}, \delta(s, pos_{vis}(n^k)) + \sigma)$, where $pos_{vis}(n^k)$ is the the $k$'th closest visible position of any covering node to $s$. Each node in $s_{NNC}$ computes a sensing probability $p_{sensing} = \frac{k}{|s_{NNC}|}$. For $\sigma = 0\,m$, $s_{NNC}$ includes exactly the $k$ nearest neighbors of $s$. For larger values of $\sigma$, $s_{NNC}$ grows to increase the probability that the true nearest neighbors are included, even if other nodes erroneously report closer positions. We thereby exploit a property of the location uncertainty: Large errors in positioning are less likely than small errors, thus it is unlikely that a node at a very long distance is included in $s_{NNC}$. The complete algorithm is shown in Fig. 3.

A major problem of this approach is that there is a significant probability $p_{\emptyset} = (1 - \frac{k}{|s_{NNC}|})^n$ that no node will take a sensor reading, e.g., for $k = 1$ and $|s_{NNC}| = 3$, $p_{\emptyset} \approx 30\%$. Thus, the overall coverage is reduced. The reason for this is that the decision for taking a reading is statistically independent on all nodes. While we could reduce $p_{\emptyset}$ by increasing $p_{sensing}$, this would also increase the amount of redundant sensor readings taken.

```
Require: Query $Q$ = ($S$, $t$, $k$, $\delta_{max}$)
  $pos_{vis}(r) \leftarrow$ GPS fix
  $s \leftarrow \{s | s \in Q.S \wedge \delta(s, pos_{vis}(r)) \leq Q.\delta_{max}\}$
  $broadcast(s, \delta(s, pos_{vis}(r)))$
  $\Delta \leftarrow \emptyset$
  repeat
    $receiveBroadcast(s', \delta(s', pos_{vis}(r')))$
    if $s' = s$ then
      $\Delta \leftarrow \Delta \cup \delta(s', pos_{vis}(r'))$
    end if
  until timeout $t_0$ is reached
  $\delta_{limit} \leftarrow min(\delta_{max}, min_k(\{\delta | \delta \in \Delta\}) + \sigma))$
  if $\delta(s, pos_{vis}(r)) \leq \delta_{limit}$ then
    $s_{NNC} \leftarrow \{\delta | \delta \in \Delta \wedge \delta \leq \delta_{limit}\}$
    $p_{sensing} = \frac{Q.k}{|s_{NNC}|}$
    return $read(t)$ with probability $p_{sensing}$
  end if
```

Figure 3. Independent Nearest-Neighbor Candidate (NNC) selection scheme. $min_k$ denotes the $k$'th smallest value from the set.

```
Require: Query $Q$ = ($S$, $t$, $k$, $\delta_{max}$)
  $pos_{vis}(r) \leftarrow$ GPS fix
  $s \leftarrow \{s | s \in Q.S \wedge \delta(s, pos_{vis}(r)) \leq Q.\delta_{max}\}$
  $broadcast(s, \delta(s, pos_{vis}(r)))$
  $\Delta \leftarrow \emptyset$
  repeat
    $receiveBroadcast(s', \delta(s', pos_{vis}(r')))$
    if $s' = s$ then
      $\Delta \leftarrow \Delta \cup \delta(s', pos_{vis}(r'))$
    end if
  until timeout $t_0$ is reached
  if $\delta(s, pos_{vis}(r)) \leq min_k(\{\delta | \delta \in \Delta\})$ then
    return $read(t)$
  end if
```

Figure 4. Deterministic Nearest-Neighbor (DNN) selection scheme. $min_k$ denotes the $k$'th smallest value from the set.

To alleviate this problem, we introduce a deterministic nearest-neighbor selection selection scheme called DNN. For DNN we choose the nodes that report the $k$ smallest distances $\delta(s, pos_{vis}(r))$ instead of computing a random selection. The complete algorithm is shown in Fig. 4. By directly using the reported position of nodes, DNN will be subject to the same influence of location uncertainty as the naive approach. It will however provide a better coverage compared to NNC.

## *B. Coordinated Selection Schemes*

Another approach to reduce $p_{\emptyset}$ is to remove the statistical independence of the sensor selection. By introducing coordi-

```
Require: Query $Q$ = ($S$, $t$, $k$, $\delta_{max}$)
  $pos_{vis}(r) \leftarrow$ GPS fix
  $s \leftarrow \{s | s \in Q.S \wedge \delta(s, pos_{vis}(r)) \leq Q.\delta_{max}\}$
  $t \leftarrow \frac{\delta(s, pos_{vis}(r))}{Q.\delta_{max}} * t_{DCNN} + jitter$
  $cnt \leftarrow 0$
  repeat
    $receiveBroadcast(snm(s'))$
    if $s' = s$ then
      $cnt \leftarrow cnt + 1$
      if $cnt = Q.k$ then
        Abort timeout and discard query
      end if
    end if
  until timeout $t$ is reached
  $broadcast(snm(s))$
  return $read(t)$
```

Figure 5. Distance-Coordinated Nearest-Neighbor (DCNN) selection scheme

Table II
VALUES FOR THE VARIABLES CHANGED IN OUR SIMULATIONS

| | |
|---|---|
| Number of Nodes (#nodes) | 200, 500, 1000 |
| $\sigma[m]$ | 0, 1, 3, 5, 10, 30, 50 |
| Minimum distance of sensing sites ($2 * \delta_{max}$) $[m]$ | 10, 50, 100 |
| Timeout for grouping nodes ($t_0$) [s] | 3 |
| Maximum backoff for CNNC ($t_{CNNC}$) [s] | 1 |
| Maximum backoff for DCNN ($t_{DCNN}$) [s] | 0.5 |
| Node speed range [m/s] | 0.5 . . . 1.7 |

nation amongst the nodes, we ensure that at least $k$ nodes in a group will take a reading.

In the Coordinated Nearest-Neighbor Candidate (CNNC) selection scheme we first use the technique from NNC to compute $s_{NNC}$. Instead of taking a reading at a fixed probability, every nearest-neighbor candidate now chooses a backoff time uniformly at random. When the backoff timer expires, a node takes a reading and broadcasts a sensing notification message (SNM) to all other nodes in $s_{NNC}$. Upon receiving the $k$'th SNM for its group, a node aborts the backoff timer and discards the query. Fig. 6 shows the complete algorithm.

We further introduce a coordinated variant of DNN, Distance-Coordinated Nearest-Neighbor selection (DCNN). Similar to CNNC, DCNN uses a backoff timer $t$ and SNMs to decide which nodes will take a reading. However, in DCNN $t$ is chosen proportional to $\delta(s, pos_{vis}(r))$. Therefore, we do not need to exchange information about other nodes in the group prior to taking a reading. To avoid nodes with similar distances choosing similar backoff timeouts and thus causing collisions when transmitting SNMs, a random jitter is added to $t$. The complete algorithm is shown in Fig. 5.

## VII. EVALUATION

We present the methodology used for our evaluation in subsection VII-A. Subsections VII-B and VII-C present and discuss the results for quality and efficiency metrics.

```
Require: Query $Q$ = ($S$, $t$, $k$, $\delta_{max}$)
  $pos_{vis}(r) \leftarrow$ GPS fix
  $s \leftarrow \{s | s \in Q.S \wedge \delta(s, pos_{vis}(r)) \leq Q.\delta_{max}\}$
  $broadcast(s, \delta(s, pos_{vis}(r)))$
  $\Delta \leftarrow \emptyset$
  repeat
    $receiveBroadcast(s', \delta(s', pos_{vis}(r')))$
    if $s' = s$ then
      $\Delta \leftarrow \Delta \cup \delta(s', pos_{vis}(r'))$
    end if
  until timeout $t_0$ is reached
  $\delta_{limit} \leftarrow min(\delta_{max}, min_k(\{\delta | \delta \in \Delta\}) + \sigma)$
  if $\delta(s, pos_{vis}(r)) \leq \delta_{limit}$ then
    $t \leftarrow uniform(0, t_{CNNC})$
    $cnt \leftarrow 0$
    repeat
      $receiveBroadcast(snm(s'))$
      if $s' = s$ then
        $cnt \leftarrow cnt + 1$
        if $cnt = Q.k$ then
          Abort timeout and discard query
        end if
      end if
    until timeout $t$ is reached
    $broadcast(snm(s))$
    return $read(t)$
  end if
```

Figure 6. Coordinated Nearest-Neighbor Candidate (CNNC) selection scheme. $min_k$ denotes the $k$'th smallest value from the set.

### A. Methodology

We implemented our sensor selection algorithms for the OMNeT++ network simulator [31] using the INETMANET extension. For the ad-hoc WiFi communication we used the 802.11 implementation from INETMANET. To improve the runtime of our simulation we restricted the maximum WiFi communication range to 150 m. For the mobile internet connection we created a simple model of UMTS with a data rate of 386 kbps shared amongst all nodes and a delay modeled according to empirical measurements [32]. Node mobility was generated using CanuMobiSim [33] on a $1\,\text{km}^2$ street graph fragment of the city-center of Stuttgart.

To create uncertain position fixes, we generate an offset by picking a direction uniformly at random and drawing a distance from a normal distribution with standard deviation $\sigma$. This offset is then added to the real position of a node.

Our simulations ran for roughly 10 simulated minutes each. In all simulations we introduced about one query per minute, yielding 10 queries per simulation. At most one query was active at any given time during a simulation run. Sensing sites were placed on roads only. The parameters chosen for our simulation are presented in Table II. For every set of parameters we repeated the simulation ten times.

Simulations with a varying number of nodes and different values for $\delta_{max}$ showed the expected behavior: Coverage goes

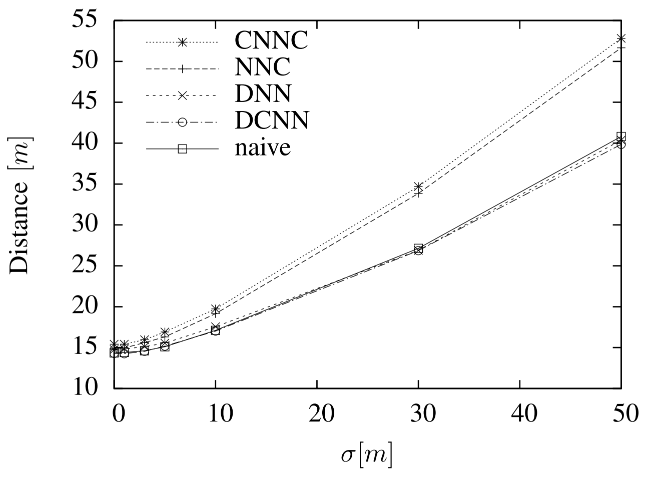

Figure 7. Mean distance of true reading positions to sensing sites

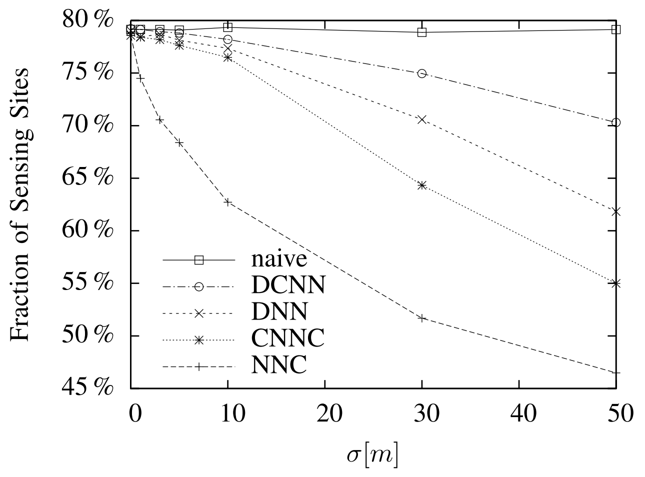

Figure 8. Mean fraction of covered sensing sites

down when there are fewer nodes and the mean distance goes up when $\delta_{max}$ grows. Since these results are obvious, we will not consider different node numbers and $\delta_{max}$ values in the following simulations, and use the following values instead: #nodes $= 1000$, $\delta_{max} = 100\,m$. $k$ was fixed to 1.

## B. Quality of Query Results

The first quality metric we consider is the distance of readings to sensing sites. To get a more accurate impression of how well our algorithms perform, we calculate the distance of each reading based on its true position $r_{true}$ rather than the visible position that would be recorded in a real system. Our measurements are shown in Fig. 7. Note that each data point is averaged over all readings returned for any query of the corresponding parameter set. The average distance of the naive algorithm for $\sigma = 0\,m$ defines the optimum average distance of $14.4\,m$ for our scenario.

The overall increase in average distance of all algorithms is a direct consequence of location uncertainty. With a growing error in positioning, the actual distance of the node reporting to be closest to the sensing site may increase for varying position uncertainties $\sigma$. The deterministic algorithms, which always use the reading that was reportedly taken closest to the sensing site – naive, DNN and DCNN – yield the smallest distance for any $\sigma$. For the probabilistic approaches the average distance of readings quickly increases as $\sigma$ grows. Since the nearest-neighbor set grows with $\sigma$, these approaches basically select any covering node at random for large $\sigma$.

Second, we look at the coverage of queries. Fig. 8 shows our measurements for the coverage metric. Each data point is averaged over all queries for the corresponding parameter set. The naive algorithm again defines the optimum coverage at a stable percentage of 79%. Thus 21% of sensing sites were not covered by any node.

The coverage of all our algorithms degrades compared to the naive approach as $\sigma$ increases. Apart from NNC, the reason for this degradation is that for the coverage we did not include readings where $\delta(r_{true}, s) > \delta_{max}$. As the location uncertainty

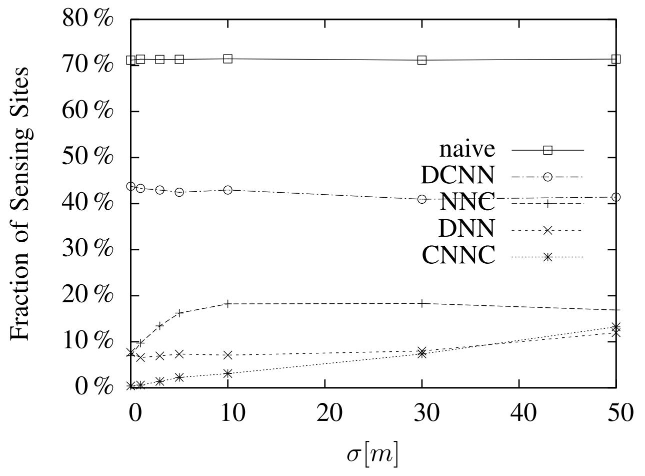

Figure 9. Fraction of sensing sites where more than one reading per query was taken

increases, there is a chance for taking a reading at a distance larger than $\delta_{max}$, which is what happens for DNN. DCNN improves over DNN because, as we will see later, it does take a large amount of redundant readings. It is, therefore, likely that at least one of the readings for each sensing site was taken at a distance smaller than $\delta_{max}$. The coverage of CNCC is even lower than that of DNN since CNNC generally picks readings at a larger distance, as can also be seen in Fig. 7. NNC clearly shows the worst coverage. As explained in Section VI-A, there is a significant probability $p_{\emptyset}$ that no node in a coverage group will take a reading. This probability grows with the number of nodes to pick from. This grows with $\delta_{limit}$, which, in turn, is dependent on $\sigma$.

## C. Efficiency of Query Execution

To analyze the efficiency of our algorithms, we use three performance metrics: The number of redundant readings, the sensor load, and the network load.

The first measure of efficiency we consider is the amount

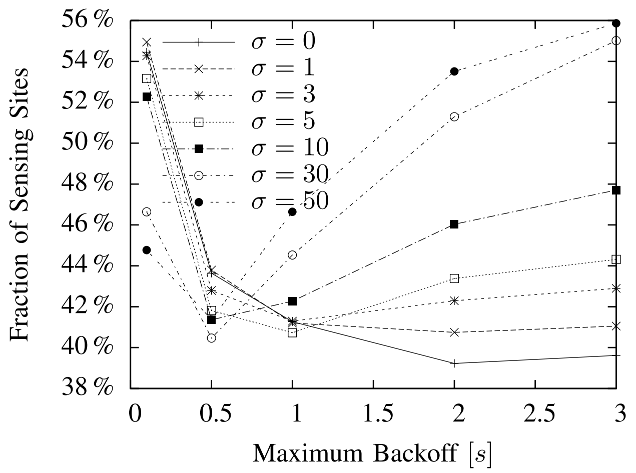

Figure 10. Influence of maximum backoff time on the fraction of sensing sites with redundant readings in DCNN.

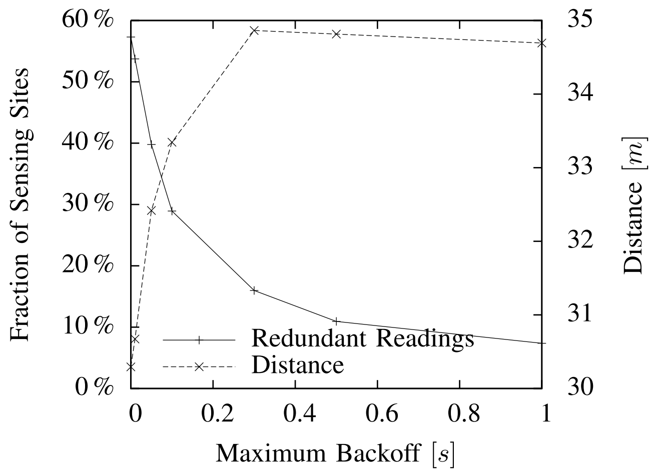

Figure 11. Influence of maximum backoff time on distance and redundant readings in CNNC, $\sigma = 30\,m$. Left y-axis: Percentage of sensing sites with redundant readings. Right y-axis: Average Distance of readings to sensing sites.

of redundant sensor readings. Redundant readings occur when there are more than $k$ readings taken for a sensing site in a single query execution. They increase the energy consumption on nodes and put additional load on the network for transmitting these readings, thus an algorithm taking fewer redundant readings is more efficient. Fig. 9 shows the fraction of all sensing sites where more than one reading was taken for a query execution. As we set $k = 1$ in our simulations, each additional reading that was taken is redundant. The naive algorithm shows redundant sensor readings at about 70% of sensing sites which is the worst case. Comparing this to the results for the coverage metric in Fig. 8 we see that about 9% of sensing sites were covered by exactly one node.

Of all our optimized approaches DCNN clearly shows the largest fraction of sites with redundant readings. The cause of this is the selection of the backoff time $t_{DCNN}$. As $t_{DCNN}$ is directly proportional to the reported distance, nodes with a similar reported distance will choose $t_{DCNN}$ so that they will obtain a sensor reading before overhearing each others notification messages. Fig. 10 shows an analysis of the amount of sensing sites with redundant readings for different backoff times in DCNN. Increasing the maximum backoff from $0.1\,s$ to $0.5\,s$ reduces the amount of sites with redundant sensings by up to 20%. Increasing the maximum backoff further is beneficial only in cases of low position uncertainty.

DNN shows an almost constant amount of sensing sites with redundant readings. Redundant readings are due to nodes having different views on the number of nodes $n'$ and the distances $\delta(s, pos_{vis}(n'))$ in their coverage group $s_{NNC}$, most likely caused by message collisions during the grouping phase.

Redundant readings in NNC are caused by multiple nodes choosing to take a reading for the same sensing site due to the statistical independence of the nodes decisions. Looking at CNNC we can see that coordination improves this problem, as the overall fraction of sensing sites with redundant readings is lower. Redundant readings in this case are caused by nodes choosing similar backoff times. This is shown in detail in Fig. 11. If the maximum backoff time is increased, the chance for two nodes choosing a similar backoff time is drastically reduced, and, in turn, the number of redundant readings is also reduced. Increasing the maximum backoff time does, however, have a negative impact on the average distance of readings and, therefore, the quality of the result, as nodes have more time to move away from their corresponding sensing site.

Next, we look at the sensor load. Sensor load is defined as the number of sensor readings a mobile node has acquired during the whole simulation. It is used as an indicator for the amount of on-device resources that are used during query execution, e.g., energy for sampling the sensor and CPU time for processing the reading. Similar to the number of redundant readings, the naive algorithm also shows the highest sensing load (see Fig. 12). About 90% of nodes had to acquire a reading 10 times. As we introduced 10 queries in each simulation and set $k = 1$, we can conclude that 90% of nodes took part in every query. All of our approaches show a much lower sensing load. In our approaches, 90% of nodes took part in only one query except for DCNN, where 95% of nodes took part in three queries. The large amount of redundant sensor readings taken by DCNN does also show in the sensor load. Since more readings are being taken per query, each individual node also has to take more readings.

Last we look at the network load. Network load is defined as the number of messages (WiFi and UMTS) sent by each node (see Fig. 13). Values are cumulated over all simulations where $\sigma = 0\,m$. By comparison, the naive algorithm causes the highest network load, where 90% of nodes had to transport up to 250 messages each, as a reading is reported for every single node in the system. In each of our algorithms the network load was reduced to less than 200 messages for 90% of the nodes, thus yielding a 20% increase in efficiency. Also notice that under the naive approach nodes had to transport up to 1300

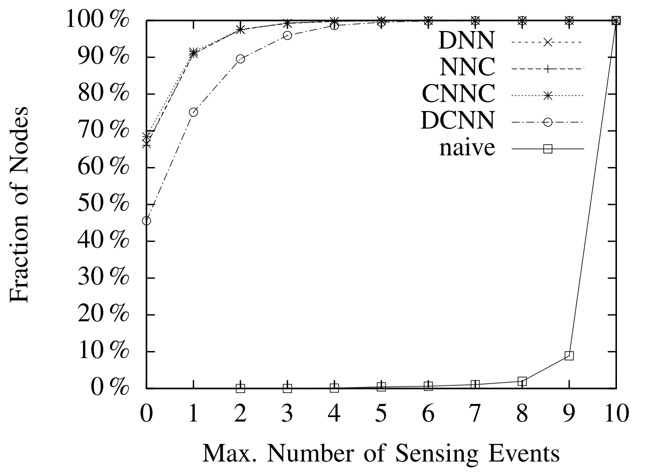

Figure 12. Cumulated sensor load. Values are cumulated over all repetitions where $\sigma = 0\,m$. Plot shows fraction of nodes where the number of sensing events $<$ maximum number of sensing events.

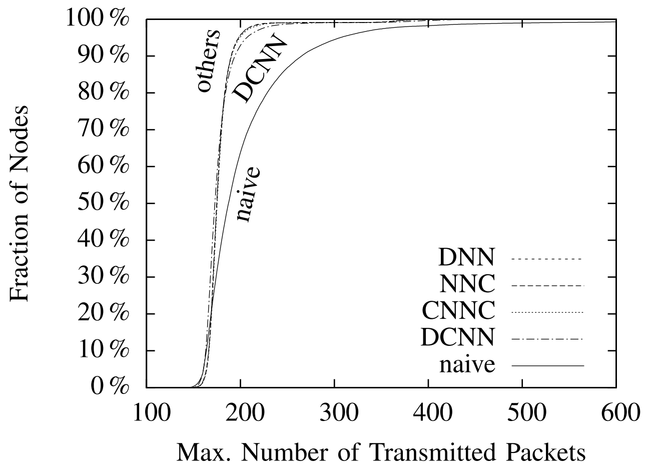

Figure 13. Cumulated number of packets sent. Plot is truncated at 600 packets. Values are cumulated over all repetitions where $\sigma = 0\,m$. Plot shows fraction of nodes where num. packets $<$ maximum num. packets.

messages whereas in our approaches no node had to transport more than 550 messages. We can see that reducing the number of redundant sensings does also pay off w.r.t. network load. The increased number of messages required for grouping and coordinating nodes is easily compensated by the reduced effort for collecting and transmitting the resulting readings.

We performed a brief analysis of energy consumption for communication, using empirical energy models [34], [35]. We can see from Fig. 14 that as with the network load, the added messages for grouping and coordinating nodes are fully compensated by the reduced number of transmitted results. Currently, the energy savings resulting from our algorithm are relatively small. This is due to the ring-based routing algorithm that we use, which in itself produces duplicate messages. In the future, using a more efficient multi-hop-routing algorithm will allow our algorithms to reduce energy usage even further.

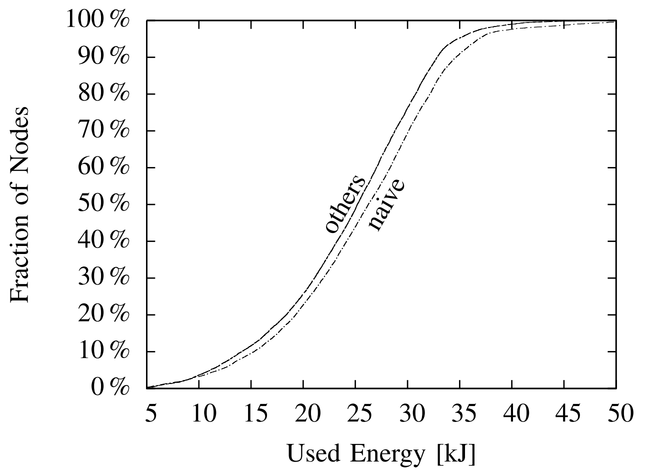

Figure 14. Cumulated Energy Consumption for WiFi and UMTS. Plot shows fraction of nodes where energy usage $<$ total energy used. Image is truncated at $50\,kJ$.

Our evaluation showed that the distance-coordinated nearest-neighbor algorithm (DCNN) yields the same average distance as an algorithm using global knowledge and shows 88% coverage in the worst case compared to the optimum value. However, the independent deterministic nearest-neighbor (DNN) performs better than DCNN under all efficiency metrics, e.g., up to 80% reduction of redundant readings over DCNN, while still providing 78% of the optimal coverage in the worst case. The approaches based on nearest-neighbor candidates show a significant reduction in quality, however, they show an improved efficiency over the corresponding nearest-neighbor approaches. In more detail, the evaluation showed that the proposed set of algorithms allows the user to define the trade-off between quality and efficiency by choosing a suitable algorithm from the proposed ones.

## VIII. Conclusion

In this paper we presented a flexible sensor network abstraction layer for general public sensing systems. We showed how the design of a sensor network can be converted into queries for a public sensing system using a virtual sensor concept.

To this effect we developed four optimized algorithms to select mobile nodes for executing the query in the face of uncertain location information. We analyzed the performance of our algorithms subject to metrics for result quality and efficiency. Moreover, we proposed different algorithms to collect sensor readings for these virtual sensors such that the quality of the result is maximized while the necessary effort w.r.t. redundant sensings, number of readings, and communication is minimized. The evaluation showed that the proposed set of algorithms allows the user to trade-off an increase in quality for a decrease in efficiency by choosing a suitable algorithm.

In the future, our work may be extended by lifting the assumption that all nodes in a coverage group can communicate directly to each other, thus allowing for larger values of $\delta_{max}$. We also plan to evaluate the impact of location uncertainty on a number of different environmental variables, e.g., light, sound, temperature, or air pollution. Finally, we plan to test our algorithms in a real-world deployment.

## Acknowledgments

This work was partially funded by the SpoVNet project of Baden-Württemberg Stiftung gGmbH.

## References

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

[2] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn, "The Rise of People-Centric Sensing," *IEEE Internet Comput.*, vol. 12, no. 4, pp. 12–21, 2008.

[3] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility Improves Coverage of Sensor Networks," in *MobiHoc '05: Proc. of the 6th ACM Intl. Symp. on Mobile Ad-Hoc Networking and Computing*. New York, NY, USA: ACM, 2005, pp. 300–308.

[4] T. W. R. Group. The Sensor Network Museum – Tmote Sky. Website. ETH Zurich. [Online]. Available: http://www.snm.ethz.ch/Projects/TmoteSky

[5] Skyhook, "Skyhook: How It Works," Electronic, [Online; 2010/11/12].

[6] T. Simonite. (2010, August) Bringing Cell-Phone Location-Sensing Indoors. Technology Review Online. MIT. [Online]. Available: http://www.technologyreview.com/communications/26156/

[7] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo, and A. T. Campbell, "Urban Sensing Systems: Opportunistic or Participatory?" in *HotMobile'08: Proc. of the 9th Workshop on Mobile Computing Syst. and Applicat.* New York, NY, USA: ACM, 2008, pp. 11–16.

[8] A. Steed and R. Milton, "Using Tracked Mobile Sensors to Make Maps of Environmental Effects," *Personal and Ubiquitous Computing*, vol. 12, no. 4, pp. 331–342, 2008.

[9] N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, and L. Steels, "Citizen Noise Pollution Monitoring," in *Proc. of the 10th Ann. Int. Conf. on Digital Government Research*, ser. dg.o '09. Digital Government Society of North America, 2009, pp. 96–103.

[10] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell, "Bubble-sensing: Binding Sensing Tasks to the Physical World," *Pervasive and Mobile Computing*, vol. 6, no. 1, pp. 58 – 71, 2009.

[11] E. Commission, "The Green Paper on Future Noise Policy (COM(96) 540)," November 1996.

[12] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. Hansen, and M. B. Srivastava, "Evaluating Participation and Performance in Participatory Sensing," in *Int. Workshop on Urban, Community, and Social Applicat. of Networked Sensing Syst. (UrbanSense) at Sensys*, November 2008, p. 5.

[13] ——, "Using Context Annotated Mobility Profiles to Recruit Data Collectors in Participatory Sensing," in *Proc. of the 4th Int. Symp. on Location and Context Awareness (LOCA 2009)*. Springer-Verlag, May 2009, p. 18.

[14] S. Reddy, D. Estrin, and M. B. Srivastava, "Recruitment Framework for Participatory Sensing Data Collections," in *Int. Conf. on Pervasive Computing (Pervasive)*, May 2010, p. 18.

[15] u-blox ag, "GPS Navigation Performance of TIM GPS Receivers," Applicaton Note, April 2002.

[16] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "Anonysense: Privacy-Aware People-Centric Sensing," in *Proceeding of the 6th Intl. Conf. on Mobile Syst., Applicat. and Services*. New York, NY, USA: ACM, 2008, pp. 211–224.

[17] E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell, "CenceMe – Injecting Sensing Presence into Social Networking Applications," in *Smart Sensing and Context*, ser. Lecture Notes in Computer Science, G. Kortuem, J. Finney, R. Lea, and V. Sundramoorthy, Eds. Springer Berlin / Heidelberg, 2007, vol. 4793, pp. 1–28.

[18] E. Kanjo, S. Benford, M. Paxton, A. Chamberlain, D. S. Fraser, D. Woodgate, D. Crellin, and A. Woolard, "MobGeoSen: Facilitating Personal Geosensor Data Collection and Visualization Using Mobile Phones," *Personal and Ubiquitous Computing*, vol. 12, pp. 599–607, 2008.

[19] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, "People-Centric Urban Sensing," in *WICON'06: Proc. of the 2nd Annu. Intl. Workshop on Wireless Internet*. New York, NY, USA: ACM, 2006, p. 18.

[20] H. Weinschrott, F. Dürr, and K. Rothermel, "Efficient Capturing of Environmental Data with Mobile RFID Readers," in *MDM '09: Proc. of the 2009 10th Int. Conf. on Mobile Data Manage.: Syst., Services and Middleware*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 41–51.

[21] ——, "StreamShaper: Coordination Algorithms for Participatory Mobile Urban Sensing," in *Proc. of the 7th IEEE Int. Conf. on Mobile Ad-hoc and Sensor Syst. (MASS'10)*. San Francisco, CA, USA: IEEE, November 2010, pp. 1–10.

[22] P. Baier, H. Weinschrott, F. Dürr, and K. Rothermel, "MapCorrect: automatic correction and validation of road maps using public sensing," in *36th Ann. IEEE Conf. on Local Comput. Networks (LCN 2011)*, Bonn, Germany, Oct. 2011.

[23] Z. Abrams, A. Goel, and S. Plotkin, "Set K-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks," in *IPSN '04: Proc. of the 3rd Int. Symp. on Inform. Process. in Sensor Networks*. New York, NY, USA: ACM, 2004, pp. 424–432.

[24] H. Zhang and J. Hou, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, no. 1-2, 2005.

[25] A. Das and D. Kempe, "Sensor Selection for Minimizing Worst-Case Prediction Error," in *IPSN '08: Proc. of the 7th Intl. Conf. on Inform. Process. in Sensor Networks*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 97–108.

[26] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," in *VLDB '04: Proc. of the 30th Intl. Conf. on Very Large Data Bases*. VLDB Endowment, 2004, pp. 588–599.

[27] H. González-Banos, "A Randomized Art-Gallery Algorithm for Sensor Placement," in *SCG '01: Proc. of the 17th Annu. Symp. on Computational Geometry*. New York, NY, USA: ACM, 2001, pp. 232–240.

[28] A. Krause, E. Horvitz, A. Kansal, and F. Zhao, "Toward Community Sensing," in *IPSN '08: Proc. of the 7th Intl. Conf. on Inform. Process. in Sensor Networks*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 481–492.

[29] D. Budzik, A. Singh, M. Batalin, and W. Kaiser, "Multiscale Sensing with Stochastic Modeling," in *Intelligent Robots and Syst., 2009. IROS 2009. IEEE/RSJ Int. Conf. on*, 10-15 2009, pp. 4637–4643.

[30] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings, "Decentralised Coordination of Mobile Sensors Using the Max-Sum Algorithm," in *IJCAI'09: Proc. of the 21st Intl. jont Conf. on Artifical Intell.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 299–304.

[31] A. Varga. The OMNeT++ Simulator. [Online]. Available: http://www.omnetpp.org/

[32] J. M. Cano-Garcia, E. Gonzalez-Parada, and E. Casilari, "Experimental Analysis and Characterization of Packet Delay in UMTS Networks," in *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, ser. Lecture Notes in Computer Science, 2006, pp. 396–407.

[33] I. Stepanov. CANU Mobility Simulation Environment (CanuMobiSim). Institut for Parallel and Distributed Systems, University of Stuttgart. [Online]. Available: http://canu.informatik.uni-stuttgart.de/mobisim/index.html

[34] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proc. of the 9th ACM SIGCOMM Conf. on Internet Measurement Conf.*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 280–293.

[35] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. Ylä-Jääski, "Practical power modeling of data transmission over 802.11g for wireless applications," in *Proc. of the 1st Int. Conf. on Energy-Efficient Computing and Networking*, ser. e-Energy '10. New York, NY, USA: ACM, 2010, pp. 75–84.