StreamShaper: Coordination Algorithms for Participatory Mobile Urban Sensing

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

***Abstract*—In this paper we introduce mechanisms for automated mapping of urban areas that provide a virtual sensor abstraction to the applications. We envision a participatory system that exploits widely available devices as mobile phones to cooperatively read environmental conditions as air quality or noise pollution, and map these measurements to stationary virtual sensors. We propose spatial and temporal coverage metrics for measuring the quality of acquired sensor data that reflect the conditions of urban areas and the uncontrolled movement of nodes. To achieve quality requirements and efficiency in terms of energy consumption, this paper presents two algorithms for coordinating sensing. The first is based on a central control instance, which assigns sensing tasks to mobile nodes based on movement predictions. The second algorithm is based on coordination of mobile nodes in an ad-hoc network. By extensive simulations, we show that these algorithms achieve a high quality of readings, which is about 95% of the maximum possible. Moreover, the algorithms achieve a very high energy efficiency allowing for drastic savings compared to uncoordinated sensing.**

I. Introduction

In recent years mobile phones have evolved to powerful mobile sensor platforms that promise to serve as a base for creating maps of environmental phenomena. They include multiple sensors such as microphones or GPS-receivers [1], and external sensors can be easily connected to the phones [2], e.g., using Bluetooth. At the same time, the interest in contributing to detailed maps of environmental conditions is growing, as the increasing number of participants in community-based projects as OpenStreetMap [3] indicates. These developments contribute to the emerging field of public urban sensing [1], [4], which aims for geo-temporal maps created by mobile users.

Maps of environmental conditions such as noise [5] or air pollution [6] are valuable when assessing the quality of living or even health threats in urban areas. While these maps can be used by citizens to avoid, e.g., regions that are highly noise polluted, scientists and city administrations can use these maps to develop counter measures. For example, reroute traffic or impose speed limits.

The potential of public sensing comes from the large number of people that cover large urban areas while carrying their mobile phones. As [7] shows, such mobile sensors achieve a much higher coverage value compared to the same number of static sensor nodes. However, the use of these mobile sensors introduces several challenges. The first challenge is the need to track the mobile sensors in order to map their measurements. The higher the required map resolution, the higher is also the need for tracking accuracy. The second challenge is to determine the quality, i.e., coverage and accuracy, of the sensed data that comprises the map. This is crucial since mobility of sensors is uncontrolled and, therefore, no guarantees about coverage and data quality are possible. Finally, the impact of map generation may not interfere with normal operation of a mobile phone, i.e., the energy consumption for sensing and tracking of the mobile devices needs to be restricted. Although Steed and Milton [8] show that using tracked mobile sensors to make fine-grained maps of environmental effects is feasible, quality and energy considerations remain open issues.

Our goal are mechanisms for automated mapping of urban areas that provide a virtual sensor abstraction to the applications. As [9] points out, such a data-centric abstraction is needed to decouple applications from the mobile data sources. Applications specify their quality requirements for virtual sensors, i.e., for a specific location, and mobile phones assign the corresponding measurements to the data stream of the virtual sensor of their location.

Although coverage metrics are available for wireless sensor networks to specify the quality of sensing, these metrics are not applicable to urban scenarios with mobile sensors that move along roads of urban areas. Therefore, our contribution comprises spatial and temporal coverage metrics that are especially suited for our scenario. Moreover, this paper contributes a centralized and a distributed algorithm for providing measurements with defined quality according to these metrics, while minimizing the energy consumption of mobile nodes. In essence, these algorithms schedule physical sensing according to virtual sensors' requirements and, hereby, shape the resulting stream of measurements. Finally, in extensive simulations, we show the effectiveness and efficiency of these algorithms.

In the remainder of this paper, we present our system model (Section II), followed by the model of virtual sensors (Section III). Then, we present metrics for measuring the quality of a virtual sensor's output stream (Section IV), before we present the algorithms (Section V) for coordinated sensing with defined quality. Afterwards, we present evaluation setup and results (Section VI). Finally, we present some related works (Section VII), before we conclude this paper and give a brief outlook on future work (Section VIII).

To appear in: Proceedings of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'10), San Francisco, CA, USA, Nov. 2010.

II. System model

Our system consists of a *reader network*, and a *central server*. Readers form a network and cooperatively perform sensing operations. They forward the measurements to their associated central server where these measurements are stored. Next, we describe these components and the underlying assumptions in detail.

The reader network consists of mobile nodes that are carried by mobile users moving in the service area. Although their movement is restricted to the street network, it is not controlled by the system. Nodes have integrated sensors for environmental sensing and a GPS receiver, which allows to derive trajectory information. Since the number of GPS receivers in mobile devices as smart phones is constantly growing, we assume that a large number of mobile devices can participate in the system. For ad-hoc inter-node communication, nodes have a wireless communication interface, e.g., 802.11bg. In Section V-B we will show how this ad-hoc network is used for the distributed cooperative coordination of sensing. The transmission range of this interface is denoted by r_{tx}. In addition, we assume that nodes can communicate with the central servers in the infrastructure using, for instance, GPRS or UMTS. In particular, GPRS/UMTS is used for transmitting measurements from the mobile nodes to the central server.

Mobile nodes are equipped with physical sensors. We refer to the output of a sensor as measurement M_i. $M_{i,time}$ is its acquisition time; $M_{i,pos}$ is the position of its acquisition. In order to assign measurements to positions on road segments, we use the mobile device's GPS receivers and map matching techniques such as [10]. We assume such a sensor to return a measurement instantly when accessed. Note that this assumption is valid for a wide range of sensors, e.g., thermometers or RFID readers. Furthermore, we assume that a sensor has a defined maximum read range S_{range}. In case of an RFID reader, this value is several meters and can be derived from data sheets. In contrast, in case of a temperature sensor, an application specifies this value according to its spatial resolution requirements.

The central server is an infrastructure-based node responsible for managing the measurements of the mobile readers in its service area. Moreover, it stores profiles (cf. Section III) of the virtual sensors in its service area. Further, the central server coordinates sensing of mobile readers as explained in Section V-A.

III. Virtual sensor

A virtual sensor V is a data-centric abstraction that allows to access measurements of a certain type V_{type} without the need to refer to physical IDs or addresses of readers. A virtual sensor is logically associated with a line segment $V_{segment}$, i.e., a part of a road (cf. Figure 1). Measurements of type V_{type} acquired along this segment are assigned to the corresponding virtual sensor.

A virtual sensor outputs a stream of virtual readings. A virtual reading is a collection of measurements acquired by physical sensors with well-defined spatial and temporal scope.

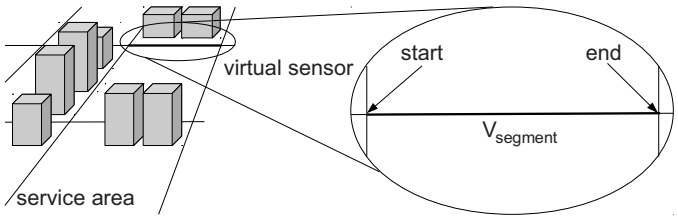

Fig. 1. Virtual Sensor

In detail, the measurements of a virtual reading fulfill the following requirements. The position $M_{i,pos}$ of each measurement is on $V_{segment}$. Moreover, the maximum temporal distance between any pair of measurements of a virtual reading does not exceed $V_{duration}$, which, in analogy to a physical sensor, is the maximum duration for the virtual sensor to acquire a virtual reading. $V_{duration}$ can also be interpreted as the maximum tolerable time difference between measurements of one virtual reading to be considered a snapshot of $V_{segment}$. Since in general multiple measurements are required to cover the segment, such snapshots cannot be acquired instantly. A virtual reading is temporally mapped to the point in time when its acquisition is finished.

We say a virtual reading covers a segment if the distance of every point on the segment to at least one measurement is smaller or equal to S_{range}. This reflects the fact that a measurement might not capture effects beyond the range of the respective sensor. For instance, an RFID reader is limited to a few meters.

In some situations, an application might desire more than one measurement for each location of a segment. For instance, to cope with inaccurate and possibly uncalibrated sensors, redundant measurements at a certain location are necessary to allow for quality-improving sensor fusion. In case of an audio sensor, e.g., a high redundancy is needed to allow for filtering of noise peaks. Therefore, we apply the concept of k-coverage. A point is k-covered, if there are k measurements within distance S_{range}. We say, a segment is k-covered if every point of it is k-covered. The value of V_k specifies the application requirement regarding k-coverage of virtual readings.

In analogy to a physical sensor, a virtual sensor has a sampling rate with a sampling interval of V_{sample}. Figure 2 shows the relation between $V_{duration}$ and V_{sample}.

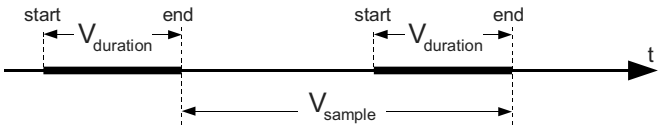

Fig. 2. Temporal bounds of a virtual reading

A profile of each virtual sensor is stored at the central server and disseminated to mobile readers. Such a profile includes the associated road segment $V_{segment}$, the type of the sensor V_{type}, and the requirement parameters regarding its output stream: $V_{duration}$, V_k, and V_{sample}.

IV. Quality metrics

Depending on the distribution and movement patterns of mobile nodes, it cannot be guaranteed that a virtual sensor meets the performance parameters specified in its profile all the time. For instance, if no node passes through the segment of a virtual sensor, no virtual reading can be acquired. In order to quantify the actual achieved quality of virtual readings, we introduce two quality metrics for the spatial and temporal coverage of virtual readings in this section.

A. Spatial coverage

The goal of the conceived spatial coverage metric is to allow for fine-grained comparisons between the achieved coverage of a virtual reading and the requested coverage as specified in the profile of a virtual sensor. Since in particular coverage "holes" are critical for applications, partial loss of coverage on a segment may not be outweighed by partially exceeded coverage requirements. Figure 3 shows several examples of virtual readings, their requested coverage V_k, and the achieved coverage. In the left example, measurements are close and therefore only a small part of the segment is covered compared to the example in the middle. In the right example, no part is 2-covered and fulfills the requirement $V_k = 2$. However, to allow for fine-grained comparisons, we consider the 1-covered parts as partially covered.

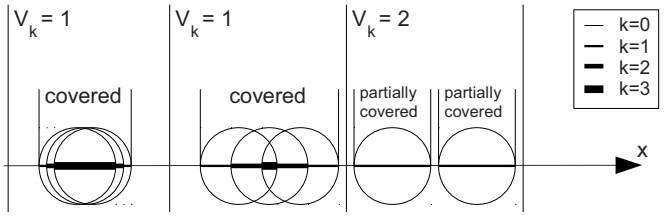

Fig. 3. Spatial coverage of a virtual reading

The achieved spatial coverage $c_s(V, R)$ of a virtual reading R of a virtual sensor V is defined as follows:

Definition 1. $c_s(V,R) = \int_{x \in V_{segment}} min(k(x), V_k) \cdot dx$

In this definition, $k(x)$ defines the coverage of a point x on the segment of V, i.e., the number of readings that cover it. Note that in the best case, when the requested coverage is fulfilled, $c_s(V, R) = V_k$. However, the achieved spatial coverage cannot exceed V_k.

B. Temporal coverage

The goal of the temporal coverage metric is to compare the achieved sampling rate of a virtual sensor with the requested sampling rate based on V_{sample}. If a virtual reading does not fulfill the spatial coverage requirements specified by the application, the sampling rate of valid virtual readings will be lower than the requested rate V_{sample}. We say a virtual reading R is invalid, if its actual spatial coverage is lower than an application specified threshold c_{th}.

Definition 2. $c_s(V, R) < c_{th} \leftrightarrow R$ *is invalid*

In order to quantify the effective reduction of the sampling rate, we introduce the temporal coverage, which is defined as follows:

Definition 3. *A point in time is temporally covered if at least one valid virtual reading precedes this point by at most* V_{sample}.

Definition 4. *The achieved temporal coverage of a set of virtual readings* R *is the ratio of covered points of time and the whole monitored time period*

V. Algorithms

In this section we describe our approach to implement the concept of virtual sensors based on mobile readers. With this approach we aim for two goals. First, we want to provide applications with sensor data that satisfies the quality requirements according to spatial and temporal coverage.

Second, we want to perform sensing as efficiently as possible. In particular, we want to avoid unnecessary sensing operations and communication operations to save energy. More detailed, we want to avoid sensing at times and places that does not increase quality or that would exceed the quality requirements.

An approach to access measurements of a specific location would be to reactively send a query as a geocast message to the readers close to that location. These nodes then could sense while they are at that location and send their measurements as reply. However, this approach has a couple of disadvantages. First, it introduces a potentially high delay since readers might take a while until they enter that location. Second, no readers might be at that location or even close to it, i.e., no measurement could be acquired. Therefore, our approach is to proactively sense while readers are on the segment of a virtual sensor, and collect these measurements in the infrastructure where applications can access it.

In detail, our approach is based on node selection schemes. We propose two schemes to adaptively select nodes, based on $V_{duration}$ and V_k, for acquiring a virtual reading: centralized spatial shaper (CSS), and distributed spatial shaper (DSS). With shaping we refer to the process of forming the output stream of a virtual sensor. Furthermore, we propose an algorithm, temporal shaper (TS), that determines when a virtual reading needs to be acquired. Finally, we propose a mechanism that derives a physical sampling interval for readers according to the maximum read range S_{range} of a physical sensor. In the following sections these algorithms are described in detail.

A. Centralized spatial shaper

The goal of this algorithm is to determine and schedule those readers for sensing that are needed to fulfill the coverage requirements, while minimizing the energy consumption of nodes. This is done by a central instance.

A simple approach to select readers for sensing during a period $V_{duration}$ would be to select all readers on a segment. Such a selection mechanism would yield the maximum possible coverage. However, this results in redundant measurements if more than V_k trajectories of readers overlap at a

specific location. An improvement to avoid the redundancy that exceeds V_k, would be to deactivate sensing of a node as soon as V_k trajectories of nodes overlap. However, position inaccuracy results in a small overlap of trajectories at such meeting points between readers. Therefore, and to reduce coordination overhead of readers, we aim for reducing the number of readers sensing on a segment.

Therefore, the idea of spatial shaping is, during a period of $V_{duration}$, to select the minimum subset of nodes on $V_{segment}$ for sensing that can achieve the coverage requirements (cf. Section V-A1). Formally, we search a minimal subset S of readers whose combined projected coverage $\int_{x \in V_{segment}} min(\sum_{s \in S} k_s(x), V_k) \cdot dx$ equals the projected spatial coverage $c(V, R)$ of the complete set of nodes R. In this equation, $k_s(x)$ is the coverage of a point x on the segment of V achieved by reader s, i.e., it is one if reader s covers it, and zero otherwise.

Then, neighboring readers are coordinated to prevent redundant sensing along overlapping trajectories (cf. Section V-A2). Moreover, progress of readers is monitored and, if necessary, other readers are selected to take over sensing (cf. Section V-A3). In the following we present a centralized algorithm for spatial shaping. A distributed algorithm (DSS) where readers cooperate is presented in Section V-B.

1) Minimum subset selection: Based on its current speed and the assumption of a uniform movement on the current segment a node can predict the part of a segment it can cover within a period of $V_{duration}$. We refer to such a part as fragment f. Each fragment is defined by a start point f_{start} and an end point f_{end}. Both are one-dimensional coordinates relative to the start point of $V_{segment}$. Note that we rely on one-dimensional coordinates to simplify the presentation of the algorithm.

The polynomial algorithm for determining the minimal subset of readers for sensing can be described as follows. At the beginning of a virtual reading, a central instance queries all nodes to predict their fragment of the segment they can cover. If the fragment's length is larger than zero, they reply with their fragment and the respective prediction. Based on all relevant fragments, the minimum subset is computed according to the Minimum Subset Selection (MSS) algorithm, as described in Figure 4.

The function $minEnd(S, k)$ returns the right border of the rightmost part of the segment that is k-covered by the fragments in S. If there is no such part, it returns zero. The method $remove(T, x)$ removes all fragments f from T where $f_{end} < x$. The function $popMaxEnd(T)$ removes the fragment f from T with largest f_{end} and returns it.

Basically, the MSS algorithm selects the longest fragments that allow to achieve the required coverage. Note that also fair strategies can be implemented that consider, e.g., the remaining energy level of nodes for the MSS algorithm. However, here we focus on effectiveness, i.e., fulfilling the coverage requirements. Therefore, the MSS first sorts the fragments according to f_{start} in ascending order. Fragments that are equal in this value, are sorted based on f_{end} in descending order. The algorithm iterates over the ordered fragments and checks for each whether the part left of it is k-covered. If this is the case, the fragment is added to a temporary set T from which it can be selected afterwards if needed. Such a case arises if the part left of the current fragment is not covered. Then, the fragment f with largest f_{end} is selected from T and added to S. This is repeated until the required coverage is achieved left of the current fragment or until T is empty. If T is empty but the coverage is not fulfilled, the current fragment is added to S.

Require: F, k sorted list of fragments, redundancy requirement
  $S \leftarrow \varnothing$ // selected fragments
  $T \leftarrow \varnothing$ // temporary container
  for all f in F **do**
    $min \leftarrow minEnd(S, k)$
    while not $empty(T)$ **and** $min < f_{start}$ **do**
      $remove(T, min)$
      if $size(T) > 0$ **then**
        $S \leftarrow S \cup popMaxEnd(T)$
        $min \leftarrow minEnd(S, k)$
      end if
    end while
    if $min < f_{start}$ **then**
      $S \leftarrow S \cup f$
    else
      $T \leftarrow T \cup f$
    end if
  end for

Fig. 4. Minimum Subset Selection (MSS) Algorithm

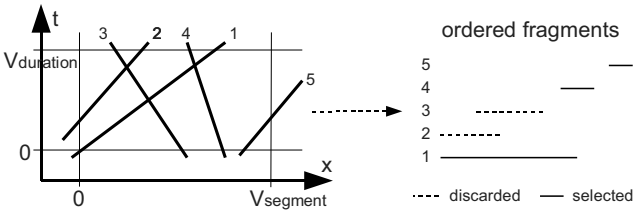

Fig. 5. Generation and ordering of fragments from predicted trajectories

The output of the MSS algorithm determines the nodes to notify to participate in sensing. Figure 5 shows the trajectories of several nodes according to their prediction and the resulting fragments. Dashed are the totally covered fragments, which can be discarded if V_k is one.

2) Reader coordination: As discussed at the beginning of this section, the meeting of two selected nodes is critical. Therefore, nodes with overlapping fragments need to coordinate to avoid redundant measurements. Basically, coordination is needed to determine for each notified reader where to start and stop sensing.

Therefore, for each node whose fragment starts at a part of the segment where the coverage requirement is exceeded, we determine its coordination partner as the node whose fragment ends at the respective part. Figure 6 shows eight selected fragments and the coverage they would achieve if they would sense uncoordinated. Assuming $V_k = 2$, the coverage

requirement would be exceeded in three parts of the segment. In the left part, (2,3) can be easily identified as coordination partners that need to avoid duplicate sensing where their trajectories overlap. Similarly, (7,8) are partners in the right part of the segment. More interesting is the middle part, where (1,5) and (4,6) are selected as partners. Alternatively, (1,6) and (4,5) can be selected as partners.

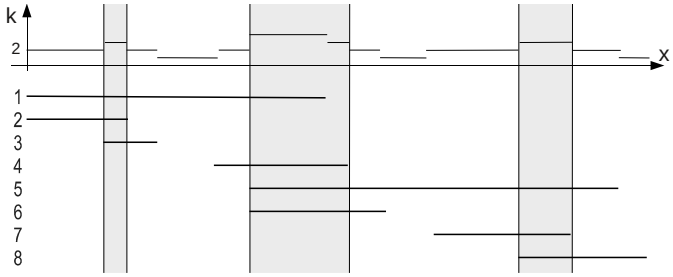

Fig. 6. Determination of coordination partners with $V_k = 2$

If a node has no coordination partner, start and stop positions are the predicted ones. Otherwise, coordination partners need to coordinate sensing where their trajectories overlap. Figure 7 shows a classification of the coordination problem, depending on movement patterns of nodes.

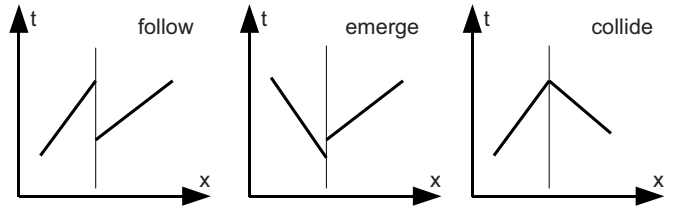

Fig. 7. Classification of Coordination Problems

The first class is *follow*, where one node follows another. In this case nodes simply start to read when notified by the central instance. However, the following node needs to stop sensing at the start point of the leading node. The second class is *emerge*. In this case two nodes pass by each other. Therefore, start coordination is necessary. One reader immediately starts to sense and the other starts as soon as it passes the point where the first reader started. We say, the second reader emerges from a covered area and starts to sense. As a heuristic solution we propose to select the reader with the highest probability to cover the gap between them. The third class is *collide*. In this case, both readers start sensing immediately. However, coordination is needed to determine the point to stop sensing of both readers.

Although coordination for classes *follow* and *emerge* can be determined based on the result of the MSS algorithm, coordination in the case of *collide* is much more challenging, and requires constant monitoring of reader movement. The idea of this coordination is to prevent nodes from sensing at the same location, and to prevent readers from stopping sensing while some gap between them is still uncovered. In the following we describe this mechanism in detail.

Initially, the central server detects the need for collide coordination and informs the affected readers along with the notification to start sensing. Based on their predictions, the readers compute the collision point and consider it as point where to stop sensing. However, as a reader's movement deviates from its prediction, the actual collision point deviates from the computed. To update the prediction at the other reader with every change of a reader's speed leads to high communication overhead.

Therefore, we propose to send updates only in two cases. First, if a reader is slower than predicted, it notifies the other reader before this would stop sensing. Hereby, a new collision point can be computed, and the remaining gap between the readers is divided and re-assigned to them. Second, if a reader is faster than predicted, it updates before it passes into the section of the other reader. Hereby, a new collision point is computed and the gap between the nodes is reassigned accordingly. Both cases are depicted in Figure 8.

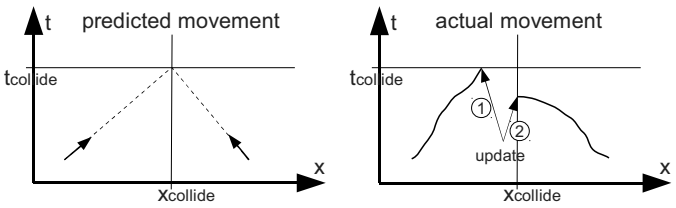

Fig. 8. Update in collide coordination

3) Progress control: Without a limitation of the period $V_{duration}$ of a virtual reading, the above described mechanisms would be sufficient for coordination. However, if a reader cannot cover its fragment within the available read period, a re-assignment of the uncovered fragment needs to be performed. For this purpose, we propose a mechanism similar to collide coordination.

According to the different coordination classes, each reader knows a fixed or variable point where to stop sensing. If its current prediction does not allow a reader to completely cover its fragment in time, it updates its prediction with the central server. Note that updates may be deferred to allow for speed fluctuations and to reduce cost for updating. However, deferring updates increases the risk for parts of the segment to remain uncovered. This is the case when a reader would have been able to take over sensing if it was informed earlier. Similarly, if a node enters a segment during $V_{duration}$ or if a node's movement deviates from its prediction in a way that allows it to cover a part of the segment that was not assigned in the MSS, it updates its prediction with the central server.

Receiving a new prediction or an updated prediction, the server then initiates the minimum subset selection algorithm on the respective part of the segment, and notifies suitable readers to start sensing. Note that the previous results of the MSS are not discarded. In contrast, nodes that were selected for sensing, are now pre-selected in the MSS.

When a reader fails that is assigned a fragment for sensing, it may not update its prediction and measurements to the

central server. To prevent this from happening, we propose an optional extension to the basic progress control mechanism that can be applied in scenarios with a high rate of reader failures. Here, readers send periodic progress reports including their measurements to the central server. If the server fails to receive a progress report it assumes the respective reader to have failed. In this case it initiates the minimum subset selection algorithm on the scope of the remaining fragment of the failed reader.

B. Distributed spatial shaper

In the previous section, a centralized spatial shaper algorithm was introduced. Although it is effective, it depends on the availability of a permanent connection between readers and a central server. Although GPRS or UMTS networks are widely available today, their energy consumption for communicating a message exceeds that of WiFi, and usage of these networks involves cost. Since the MSS algorithm only depends on information of neighboring readers and the respective fragments, we present a distributed spatial shaper (DSS) algorithm that is based on reader cooperation in the area of a virtual sensor in an ad-hoc network.

In essence, this algorithm consists of the same parts as CSS. Especially, the minimum subset selection (MSS) algorithm, as presented in the previous section (cf. Figure 4), is an essential part of the DSS. However, now we do not consider global knowledge. Instead, in an initial phase, readers exchange their predictions with neighbors. In a second phase, they cooperatively select the readers for sensing. Then, the readers coordinate sensing and perform a cooperative progress control.

1) Prediction exchange: The idea of the initial phase of DSS is to distribute local movement predictions of nodes on a segment among each other as a base for reader selection. A central server initiates a virtual reading on a segment by informing one or more readers on the segment. About details on temporal shaping see Section V-C. When a reader is informed that a new reading needs to be acquired, it propagates this information to neighboring nodes by broadcasting its prediction in the ad-hoc network. First, this triggers other readers to participate in the virtual reading. Second, it serves as a means to disseminate predictions. Every node receiving such a notification for its current segment, either from a central server or a neighboring node, updates its local view on neighboring fragments. Then, it predicts the fragment it can cover during $V_{duration}$ and broadcasts this together with the locally known fragments of other nodes. At the end of this phase every node has a local view on the fragments of nodes in the segment.

An optimization to reduce message overhead is to broadcast only if a node is selected by MSS on its local view on fragments. Since each reader broadcasts at most one message, this algorithm has, in the worst case, a linear message complexity. However, since readers suppress broadcasting if their fragment is covered, the number of messages is on average much smaller. Due to its limited message complexity, this phase is rapidly completed. We assume node movement during that time frame to be neglectable.

2) Cooperative reader selection: The idea of the second phase is that nodes achieve a consistent view on those fragments that are relevant for coordination, i.e., neighboring fragments. Based on the exchange of fragment information in the previous phase, nodes establish communication routes to nodes of other fragments based on the reverse path. Based on this routing information, nodes of neighboring fragments can communicate. Although, due to mobility, these routes break with time, communication shortly after route establishment is likely to succeed. This communication is needed to initiate reader selection.

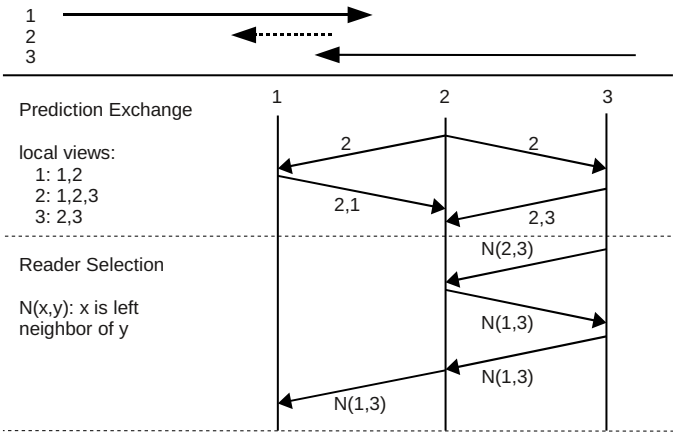

Fig. 9. Distributed cooperation: prediction exchange and reader selection

Due to possible inconsistent local views of the nodes, the goal of this phase is to consistently select readers. Therefore, each reader that locally decided to participate in sensing, contacts the reader of its left neighbor fragment. Note that the definition to contact the left neighbor is arbitrary, and does not affect performance of the mechanism. When a reader contacts its partner, it assumes, based on its local knowledge, both readers are selected for sensing. However, due to partial local knowledge, this might be false. If the receiver locally can determine that the sender or itself is not selected for sensing, it notifies the sender accordingly by replying with its local knowledge about fragments. This process is repeated until each neighbor relation is verified. When this is achieved, consistent reader selection is achieved, and each reader knows its left neighbor fragment.

Figure 9 shows an example with three nodes on a segment, where, if $V_k = 1$, only readers 1 and 3 need to sense. In the example, reader 2 initiates the prediction exchange. Nodes 1 and 3 receive the fragment of reader 2. They also broadcast the fragments of their local view, and reader 2 learns from fragments 1 and 3. In the reader selection phase, reader 3 is the only one that has a left neighbor based on its local view. It contacts this neighbor (reader 2), which however computes that it does not need to coordinate with reader 3. It replies with its local view. Based on this reply, reader 3 then again

determines its left neighbor and contacts it to initiate collide coordination.

3) Cooperative reader coordination: The idea of the reader coordination is to avoid redundant sensing due to overlapping fragments, which is needed between readers of neighboring fragments. It is performed according to the classification in Section V-A. In case of follow and emerge, coordination partners determine start and end points for sensing based on the consistent view on their fragments after reader selection.

In contrast, collide coordination requires constant monitoring of movement. Since routes may break due to movement, a reader addresses its prediction update directly to the coordination partner. The idea of this is that coordination is only needed if neighboring nodes are close. If the coordination partner is in transmission range it receives the message, computes a new meeting point and replies. Otherwise, the delivery of the update is not crucial, since there is no risk of overlapping trajectories. However, in this case, the sender needs to resend an update regularly. The sender derives the interval based on an estimation of the receiver's relative speed and on the communication range. Note that this approach does not require a routing mechanism.

4) Cooperative progress control: The idea of the progress control is to monitor the progress of sensing and, if this deviates from the predicted, to assign sensing tasks to other nodes. In essence, when a reader's prediction indicates that it cannot cover its fragment in time, other readers need to take over sensing. Therefore, it broadcasts its current prediction, its fragment, and its neighboring fragments to its direct communication neighbors. With this information a cooperative MSS mechanism is initiated in the scope of the fragment, i.e, nodes whose current predictions overlap the unassigned part of the segment exchange their predictions. Similarly, when a node's prediction indicates that it can cover an unassigned part of the segment or if a new node enters the segment, it broadcasts its current prediction, its fragment, and its neighboring fragments to its direct neighbors to initiate the MSS mechanism.

When a reader fails, it typically cannot initiate an MSS beforehand to let other readers take over sensing. Therefore, we propose an optional mechanism for monitoring readers based on periodic progress reports. Such a progress report includes the position and the fragment associated to the reader. It is broadcasted to the direct neighbors. A reader monitors those readers whose fragments overlap with its own. Based on the predicted current position of the monitored reader, it determines its probability to be in communication range. If it has a high probability to be in range, and if it fails to receive a progress report it assumes the other reader to have failed. In that case, it initiates a cooperative reader selection.

C. Temporal shaper

The basic idea of temporal shaping is to schedule virtual readings according to the V_{sample} parameter. This requires knowledge of the time of the last virtual reading. Several approaches are conceivable to manage this information. One approach can be to let the central server, which anyway stores the updates, notify readers close to the segment of the virtual server about the start of a new virtual reading. This approach is especially suited for scenarios where the sampling interval is rather large, thus managing the time of the last virtual reading cooperatively at the mobile nodes would be unreliable. However, alternatively, the readers can store the time of virtual readings and periodically propagate this information to other nodes in the vicinity of the sensor.

Both approaches are feasible, but in this paper we assume central temporal shaping controlled by the central server that initiates virtual readings according to the temporal shaping requirement by informing mobile nodes. For details on the cooperative approach, we refer to our previous work [11].

D. Resolution shaper

The purpose of the resolution shaper is to adjust the physical sampling interval such that S_{range} is respected. Basically, the resolution shaper needs to omit sensing if its position is within an area where it already acquired measurements. Moreover, it needs to adjust the physical sampling interval according to its speed v in a way that it covers a distance of $2 \cdot S_{range}$ between two measurements. For this purpose, a reader computes its sampling interval $\delta = 2 \cdot S_{range}/v$. Readers adjust δ with every update of their positioning system.

Since positioning has a high energy consumption, it is essential to deactivate it if not needed. In the sensing area, between start and end point of sensing, a reader needs continuous positioning. However, outside that area, it may deactivate positioning. We propose a positioning interval depending on the distance a node may move before it enters its sensing area or before it enters a different segment. Based on this distance and the maximum speed, a node can compute the time to deactivate positioning.

VI. Evaluation

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

- **CSS:** This implementation is based on the CSS algorithm (see Section V-A). A central instance coordinates sensing of mobile nodes.
- **DSS:** In contrast to the CSS implementation, DSS is based on the distributed stream shaping algorithm (see Section V-B), where mobile nodes coordinate sensing in an ad-hoc network. The central instance is only responsible for managing the data read by mobile nodes.
- **Isolated:** A simple isolated approach where all nodes sense independently. A node starts sensing when it enters a segment for which a virtual reading needs to be acquired; it stops sensing as soon as it leaves the segment or as soon as the period of the virtual reading is finished. This implementation presents the worst case for redundant sensing, but also the best case for coverage.

We implemented our algorithms using the 802.11 extension of ns-2 with the transmission range $r_{tx} = 100\,m$. The nodes

move at pedestrian speed (between $0.7\,m/s$ and $1.8\,m/s$) according to the UDEL pedestrian mobility model [12] on the street graph of a nine block section of Chicago. Movement patterns, as in reality, heavily depend on the simulated hour of the day. Movement predictions are done based on the current node speed by assuming uniform node movement along the current segment of a node. Although the simulated section is relatively small compared to the size of a city, it is sufficiently large for this evaluation since virtual sensors cover only road segments. More important as the size of the service area is the effect of node density which we evaluate in a wide range. Note that only a small fraction of the nodes moves simultaneously on the road network. Each edge of the street graph is assigned a virtual sensor. Each simulation is performed 10 times and lasts 1800 seconds. During that time, a virtual reading is acquired every 100 seconds. By default, $V_{duration}$ is 60 seconds, and V_k is 2. S_{range} is set to 0.5 m. With this small value, the performance of the algorithms is evaluated under most stressing conditions. Since the temporal shaper (cf. Section V-C) is already discussed in detail in previous work [11], we do not consider energy cost introduced by temporal shaping through notifying nodes to start a virtual reading.

TABLE I
ENERGY MODEL

Component	Energy [mJ]
GPS [13]	
Position Fix	75
RFID [14]	
Read	80
802.11b at 1 Mbps [15]	
(broadcast rate)	
Send (1000 Bit)	2
Receive (1000 Bit)	1
GPRS [16]	
Send (1000 Bit)	80
Receive (1000 Bit)	40

To measure the energy consumption of the mobile nodes we rely on the energy model given in TABLE I.

A. Effectiveness

In this section we evaluate the effectiveness of the algorithms CSS, DSS and Isolated in terms of achieved spatial coverage as defined in Section IV-A. As a performance metric, we compute the average spatial coverage $c_s(V, R)$ of the virtual readings acquired during a simulation run.

At first, we evaluate the performance of the algorithms simulating node mobility at different hours of the day. We plot the spatial coverage during the course of a day in Figure 10. In this scenario, the number of nodes in the simulation is 15000. Note that only a small fraction of these nodes moves on roads during simulation time. And this fraction depends on the hour of the day. At rush hour, e.g., at 16:00 o'clock, Isolated achieves its peak coverage value. However, as the coverage value of about 1.1 indicates, the node density is still too low to achieve the requested coverage of 2. As expected, the coverage highly varies during the course of the day. However, CSS and DSS show a similar behavior as Isolated resulting in a slightly lower coverage value. This gap of about 6% is due to the fact that Isolated starts sensing as soon as a node enters a segment. In contrast, CSS and DSS start sensing based on movement predictions. However, e.g., at a crossing, such predictions can only be determined after the node follows a road segment for some time. Therefore, during that time, sensing is deactivated, and the coverage is reduced. The coverage-loss depends on the position uncertainty, which is 5 m in our simulations. A node misses some measurements when the uncertainty is high compared to S_{range}. If the uncertainty is low, or if S_{range} is high, a node does not move far compared to S_{range} until it can start sensing. Therefore, it does not miss to capture measurements.

Figure 11 plots the achieved spatial coverage over V_k at 16:00 o'clock. As in the previous scenario, the number of nodes is set to 15000. The Isolated approach achieves the maximum possible spatial coverage. CSS and DSS achieve a coverage value that is about 5% lower. This gap is, as in the previous paragraph, due to the delay introduced by the movement prediction. However, as the figure shows, this gap is independent from V_k, i.e., CSS and DSS are effective for a wide range of coverage values. The achieved coverage value is always below the requested coverage V_k. This shows that some segments are not populated enough to be covered by nodes, while others are highly populated and a much higher coverage value as the requested could be achieved.

The same small loss of coverage can also be seen in Figure 12. Here, the spatial coverage is depicted for varying numbers of nodes. As the figure shows, this loss is independent of the node density. This shows that CSS and DSS are effective independent from the node density.

Moreover, we investigated the effectiveness of the algorithms for varying values of $V_{duration}$. The results confirm the presented results, in that DSS and CSS are also effective independent from $V_{duration}$. However, due to space limitations, we omit the respective figure.

B. Efficiency

In this section we evaluate the efficiency of the algorithms in terms of energy consumption (EC). The base to determine the energy consumption of a node is Table I. By default, we assume a sensor with a high energy consumption – an RFID reader. The EC is computed as the average energy a node spends for positioning, communication, and sensing. Since mapping requires continuous positioning during sensing, and due to the high share of positioning in the overall energy consumption, even usage of low power sensors would result in similar EC values as the following.

Figure 13 plots the average energy consumption per node and hour in the course of a day. In this scenario, V_k for each virtual sensor is set to 2, and the number of nodes in the simulation is 15000. Basically, this figure shows similarities

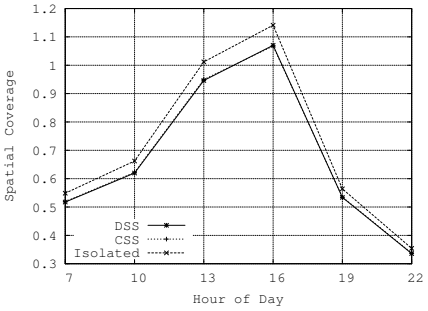

Fig. 10. c_s in the course of day

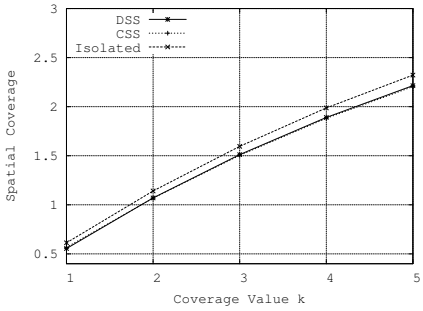

Fig. 11. c_s depending on requested coverage

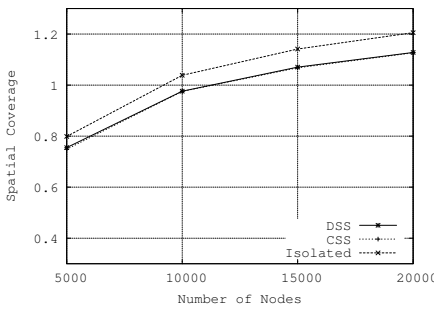

Fig. 12. c_s depending on number of nodes

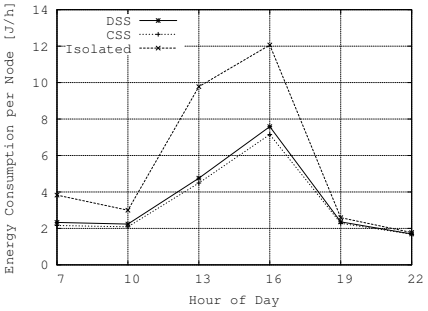

Fig. 13. EC in the course of day

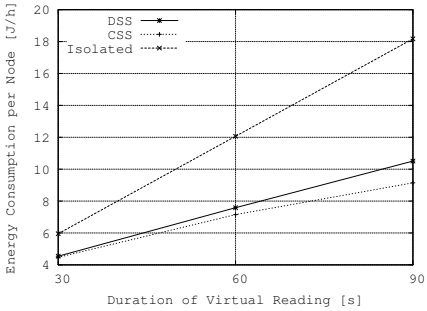

Fig. 14. EC depending on $V_{duration}$

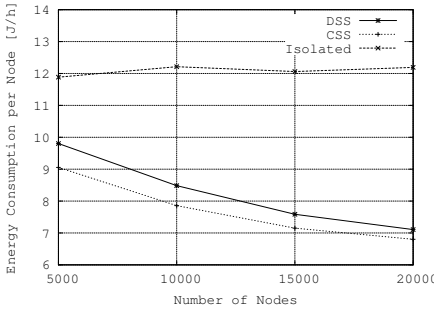

Fig. 15. EC depending on number of nodes

to Figure 10. Depending on the hour of the day, Isolated achieves different coverage values and, related to the coverage, different values for the energy consumption. However, energy consumption does not always reflect the achieved coverage. For instance, c_s at 7:00 o'clock is lower than at 10:00 o'clock, while the EC at 7:00 o'clock is higher compared to 10:00 o'clock. In essence, this is the result of different mobility patterns. At 7:00 o'clock nodes are more likely to move along the same paths. In such a scenario, a higher fraction of measurements is redundant and could be omitted. This can be seen in the EC values of DSS and CSS. Both algorithms achieve to reduce the EC by a similar amount. In case of highly redundant movement patterns of nodes, i.e., at 7:00 o'clock, the savings are almost 50%, while at 10:00 o'clock, the savings are only about 20%. At 22:00 o'clock, when only few nodes move along the roads, the possible savings are minimal. In this case, EC of all three approaches is nearly equal.

Interestingly, the energy consumption of DSS and CSS are nearly identical. While the energy consumption for communication are much higher in case of CSS due to GPRS, DSS suffers from redundant sensing. This redundancy is based on the fact that nodes can only coordinate ad-hoc if they are in transmission range.

As Figure 14 shows, this effect increases with larger $V_{duration}$. This is because nodes in DSS only participate in the coordination of their current segment.

Finally, Figure 15 plots the EC for several numbers of nodes in the network. As expected, Isolated shows an energy consumption that is independent of the node density, since each node independently senses without considering other nodes. In contrast, DSS and CSS benefit from increasing node density by preventing unnecessary sensing and, therefore, allow to reduce the energy consumption per node on average. Interestingly, the difference between DSS and CSS is decreasing with increasing node density. This is due to the improved ability of nodes for ad-hoc coordination. Moreover, with increasing node density, savings of DSS and CSS compared to Isolated increase. High node densities promise even higher energy savings for DSS and CSS compared to Isolated.

VII. Related Work

Public sensing is currently an active topic in various research fields. In the field of sensor networks, focus is mainly set on the autonomous monitoring of environmental conditions in inaccessible areas. However, projects such as [17] use sensor nodes for public sensing. Sensor nodes imply high cost for deploying dense large-scale sensor networks. Moreover, they suffer from the battery depletion problem. In our previous work [11] we propose to deploy simple and cheap RFID-based sensors at points of interest, which are then read by passing mobile devices with integrated RFID readers. Although this approaches alleviates these problems, it still depends on the deployment of sensor nodes.

Another set of approaches uses instrumented mobile devices for sensing in urban areas. Gellersen et al. [18] propose the integration of sensors into mobile devices to achieve direct context awareness of these devices. However, the locally collected sensor data is not shared between devices. In contrast, [19], [2] use mobile devices to collect shared sensor data. Rudman et al. [19] attach sensors for monitoring air pollution

to a tablet PC. MobGeoSen [2] is based on the integration of sensors to mobile phones, which are carried by a large number of people. Although [8] shows that using tracked mobile sensors to make fine-grained maps of environmental effects is feasible, these approaches do not consider quality-aware sensing and energy considerations remain open issues. One interesting approach that deals with these issues is [20]. However, it lacks distributed coordination algorithms for mobile nodes. To the best of our knowledge, our approach is the first to consider distributed coordination of sensing to achieve energy-efficient and quality-aware sensing.

Similar to our approach, [9] proposes virtual sensors as a data-centric abstraction for applications, when querying measurements of mobile devices. However, it lacks suitable metrics for measuring the quality of environmental maps of urban areas. In the field of sensor networks, a variety of coverage metrics has been proposed [21]. However, these approaches consider the coverage of planar areas with stationary or controlled mobile sensors.

VIII. Conclusions

This paper introduces mechanisms for automated mapping of urban areas that provide a virtual sensor abstraction to the applications. In a participatory system, widely available devices as mobile phones read environmental conditions as air quality or noise pollution, and map these measurements to stationary virtual sensors. We proposed spatial and temporal coverage metrics for measuring the quality of measurements and we presented two algorithms for coordinated sensing in order to achieve the quality requirements, while minimizing the average energy consumption of nodes. By simulations, we showed that these algorithms achieve about 95% of the maximum possible coverage. Moreover, the algorithms achieve a very high energy efficiency allowing for drastic savings up to 50% compared to uncoordinated sensing.

In future work we plan to include mechanisms for increasing the fairness of sensing, i.e., to consider the individual energy consumption of nodes rather than the average energy consumption. Alternatively, we plan to incorporate strategies that consider available resources of nodes.

Acknowledgements

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

References

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

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

[3] M. M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

[4] 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 Computing*, vol. 12, no. 4, pp. 12–21, 2008.

[5] N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, and L. Steels, "Citizen noise pollution monitoring," in *dg.o '09: Proceedings of the 10th Annual International Conference on Digital Government Research*. Digital Government Society of North America, 2009, pp. 96–103.

[6] S. Kim and E. Paulos, "inair: Measuring and visualizing indoor air quality," in *Ubicomp 2009*, 2009.

[7] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility improves coverage of sensor networks," in *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2005, pp. 300–308.

[8] A. Steed and R. Milton, "Using tracked mobile sensors to make maps of environmental effects," *Personal Ubiquitous Comput.*, vol. 12, no. 4, pp. 331–342, 2008.

[9] A. Kansal and F. Zhao, "Location and mobility in a sensor network of mobile phones," in *NOSSDAV 2007: 17th International workshop on Network and Operating Systems Support for Digital Audio & Video*, June 2007.

[10] A. Civilis, C. S. Jensen, J. Nenortaite, and S. Pakalnis, "Efficient tracking of moving objects with precision guarantees," in *Proc. of the First Annual Intl Conf. on Mobile and Ubiquitous Systems (MobiQuitous 2004)*, Boston, Massachusetts, USA, August 2004, pp. 164–173.

[11] H. Weinschrott, F. Dürr, and K. Rothermel, "Efficient capturing of environmental data with mobile rfid readers," in *Proceedings of International Conference on Mobile Data Management*, Taipei, Taiwan, May 2009.

[12] J. Kim, V. Sridhara, and S. Bohacek, "Realistic mobility simulation of urban mesh networks," *Ad Hoc Netw.*, vol. 7, no. 2, pp. 411–430, 2009.

[13] Navman, Apr. 2010. [Online]. Available: www.globaltime.com.cn/attachment/30DataSheet_060616105012.pdf

[14] Skyetek, Apr. 2010. [Online]. Available: www.skyetek.com/Portals/0/Documents/Products/SkyeModule_M9_DataSheet.pdf

[15] Summitdatacom, September 2008. [Online]. Available: www.summitdatacom.com/Documents/SDC-CF10G Product Brief 200803.pdf

[16] B. Gedik and L. Liu, "Mobieyes: A distributed location monitoring service using moving location queries," *IEEE Trans. on Mob. Comp.*, vol. 5, no. 10, pp. 1384–1402, 2006.

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

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

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

[20] A. Krause, E. Horvitz, A. Kansal, F. Zhao, "Toward Community Sensing," in *Proc. of Information Processing in Sensor Networks.*, Washington, DC, USA, 2008.

[21] P. Brass, "Bounds on coverage and target detection capabilities for models of networks of mobile sensors," *ACM Trans. Sen. Netw.*, vol. 3, no. 2, p. 9, 2007.