# Resource Management for Public Sensing

**Klaus Herrmann, Daniel Fischer, Damian Philipp**
University of Stuttgart, Institute of Parallel and Distributed Systems (IPVS)
Universitätstr. 38, D-70569 Stuttgart
firstname.surname@ipvs.uni-stuttgart.de

**Abstract**

Public sensing is a new research area in the fields of wireless sensor networks and mobile computing. It leverages the mobile sensors and system resources readily available in mobile phones to execute sensing tasks. In order to plan, execute and adapt large-scale sensing tasks, applications need to query for the available resources, e.g. the density of certain sensors. We investigate how such information can be provided, and we propose a resource manager for public sensing. Our primary goal is to minimize the energy consumed by the mobile devices to make public sensing feasible without disturbing users. We propose a cluster-based protocol for collecting local views of the resource state using local ad-hoc communication since this is much more energy-efficient than long-range (e.g. cellular) communication. We compare our solution to a standard approach where mobile devices communicate their resource states using the cellular phone network. We show that $65\%$ of the energy is saved and the communication load on the infrastructure is reduced by $90\%$ while an average delivery ratio of $93\%$ is retained.

## 1 Introduction

An increasing number of sensors are available in devices carried by the general population. E.g. the iPhone includes sensors to measure acceleration, brightness, proximity, sound, the magnetic field, as well as GPS and a video camera. Motivated by these capabilities, many researchers recently advocated the use of such devices for large-scale sensing tasks (Campbell et al. 2008; Cuff, Hansen, and Kang 2008; Lane et al. 2008). This is called urban, participatory, or *public sensing*. Example applications are air/noise pollution monitoring (Kanjo et al. 2009; Maisonneuve et al. 2009), large-scale discovery and classification of sound events (Lu et al. 2010), and measuring free parking lots (Lochert, Scheuermann, and Mauve 2010).

Current approaches (Parker et al. 2006; Lu et al. 2010; Cornelius et al. 2008; Kanjo et al. 2009; Maisonneuve et al. 2009) assume a very simplistic system model where the sensors and resources required for an application are always available at the area that is subject to the sensing task. However, the goals of different public sensing applications can be very heterogeneous and require gathering and processing data ranging from a local to a global scale. Since the public sensing system cannot monitor all possible data at all times, it needs information about what to measure where and in which quality depending on the application requirements. At the same time, the density, type, and accuracy of sensors may be very heterogeneous in time and space. Therefore, future public sensing applications need a planning component that configures the public sensing system depending on the currently available sensors, local computing capabilities and network properties. We call the entity providing this knowledge a *resource manager*.

Consider an application that remotely detects overcrowding in a shopping mall. This application help in planning a shopping trip such that long queues are avoided. The degree of crowding can be detected using different types of sensors. In outdoor locations, for example, GPS could be used to measure the position of users. However, this is not possible indoors, where proximity sensors (e.g. bluetooth connectivity) could be used. Both approaches are vastly different and require different detection software. The proposed resource manager would collect information about the sensors available in the respective location and provide this to the planning system that, in turn, would deploy the right detection software.

We propose the basic structure, algorithms and protocols of a resource manager that is capable of achieving this in an energy-efficient and scalable way.

Users will only participate in public sensing if this will not drain the energy of their devices, keeping them from running their own daily tasks. Therefore, our primary goal is to conserve energy. Although many current mobile devices can take advantage of cellular infrastructure connectivity (e.g. GPRS, UMTS), this type of communication consumes a lot of energy compared to ad-hoc communication (e.g. WiFi, Bluetooth) (Balasubramanian, Balasubramanian, and Venkataramani 2009). Moreover, further processing of resource data (e.g. aggregation) must be performed in the infrastructure. In order to reduce the respective load, we reduce the amount of data that reaches the infrastructure.

Our contributions can be summarized as follows:

- A cluster-based approach where few special nodes collect the resource state in their local vicinity using ad-hoc communication before sending the aggregated data to a

component in the infrastructure.

- A robust ring-based topology for in-network aggregation to cope with mobility. By dropping the assumption of time synchronization in this approach, we decrease the query execution time significantly.
- A new clustering mechanism that exploits the typical query/response communication to perform cluster maintenance, requiring only a very low overhead.
- Extensive evaluation results that show that our approach reduces energy-consumption by up to $65\%$ and that the amount of data that has to be processed in the infrastructure is reduced by up to $90\%$.

In the next section, we present our system model. Subsequently, we discuss our general approach (Section 3) and derive requirements for it (Section 4). In Section 5, we discuss the suitability of existing approaches with respect to the requirements. Then, we propose our solution (Section 6) and subsequently evaluate it in Section 7. In Section 8, we conclude the paper with an outlook.

## 2 System Model

We assume a large number of participants in the system, carrying *mobile nodes* (MNs) (e.g. mobile phones). Their incentive to provide resources for sensing is associated with the ability to access the public sensing services itself. However, there may also be a monetary compensation from commercial service providers or from public authorities.

MNs have two means for communicating: (1) ad-hoc communication (e.g. WiFi) with nearby devices forming an ad-hoc network and (2) an uplink to a cellular infrastructure (UMTS/GPRS) allowing them to communicate with dedicated public sensing nodes on the Internet. It is also conceivable that users may have intermittent opportunistic access to WiFi access points. This would add further possibilities for optimization at the price of more complex algorithms. However, it does not change the general model of a mix of wide-area and local-area communication that we will exploit.

We also assume that MNs have the ability to determine their current position. There are numerous techniques for doing this, including GPS, cell-tower/access point triangulation, or beaconing (Sun et al. 2005).

Public sensing applications submit resource queries to the resource manager. These queries request views of specific aspects (e.g. density of certain sensors), quality (e.g. maximum age) and geographical area-of-interest. Furthermore, we assume that each user specifies a period of time during which his phone must have enough energy for his daily tasks and after which he can recharge it.

## 3 Main Objectives and General Approach

Users will only participate if the public sensing system does not drain their devices' energy. Hence, we consider energy conservation as the primary requirement for the resource manager. For simple data collection and aggregation tasks, communication is the dominant factor in energy usage. However, transferring the same data packet using the cellular infrastructure requires significantly more energy than using ad-hoc communication (Balasubramanian, Balasubramanian, and Venkataramani 2009). Furthermore, cellular communication creates a large energy overhead per packet due to signaling and stand-by modes (Balasubramanian, Balasubramanian, and Venkataramani 2009). Hence, a simple approach where each MN receives and responds to application queries using cellular communications drains energy quickly. Instead, our approach is to group nodes into *clusters*, possibly spanning multiple hops. In each cluster, the *clusterhead* (CH) is responsible for collecting and aggregating the resource information of its *cluster members* using ad-hoc communication before sending the aggregate to infrastructure nodes. Only the CH communicates via the cellular infrastructure, receiving and responding to queries.

We propose a robust and flexible cluster-based aggregation scheme that has very low maintenance overhead. We take an existing clustering algorithm and integrate it tightly with the algorithms of the resource manager that are responsible for managing queries and responses. This integration gives us a big advantage over the large number of existing general-purpose clustering approaches in terms of effectiveness and efficiency. Thus, our contribution is not an entirely new stand-alone clustering approach but a clustering-based approach to public sensing.

## 4 Requirements on Clustering Algorithms

Our algorithm needs to satisfy the following requirements:

1. $r$**-hop Clusters.**: Each member always has at most $r$ hops *cluster-radius* distance to its CH. This is vital since it allows us to set a limit to the maximum delay in collecting the resource state of the members. The algorithm must allow for $r > 1$.
2. **Mobile Nodes.** The approach must be able to deal with node mobility (i.e. changing topology and partitions).
3. **Low Maintenance Overhead.** The number of cluster maintenance messages must be minimized to minimize energy consumption and delay.
4. **Scalability**. The number of messages per node must be independent of the number of MNs. This makes the system scalable.
5. **Fairness.** No individual node must be completely drained while others still have full capacity. Moreover, each participant must be able to define a limit on the amount of energy he is willing to invest in public sensing.

## 5 Related Work

Existing public sensing architectures (Parker et al. 2006; Lu et al. 2010; Cornelius et al. 2008; Kanjo et al. 2009; Maisonneuve et al. 2009) do not cope with the problem that applications have to use different plans depending on the available sensors in order to achieve their goal. They simply assume that the required sensors are in the target area and hence, do not require knowledge about the available resources. Consequently, no work on resource management in public sensing has been presented so far.

With the exception of MMAN (Kazemi, Hadjichristofi, and DaSilva 2008), existing MANET-based resource monitoring approaches (Badonnel, State, and Festor 2005; Tuduce and Gross 2004; Ramach, Belding-royer, and Almeroth 2004; chung Shen et al. 2003), assume a small scale system and operate only within a fully connected MANET. They assume that resource information is only requested locally, i.e. close to the target. MMAN can cope with partitions on a larger scale but assumes that the whole MANET is covered by dedicated monitoring devices connected out-of-band. Due to high costs, this is not feasible in the area of public sensing. Furthermore, existing approaches in this area neither focus on minimizing the energy usage nor on exploiting infrastructure connectivity.

Most clustering techniques for MANETs and sensor networks create 1-hop clusters (Abbasi and Younis 2007) in which each node has a neighboring CH. This violates Requirement 1. Some algorithms (Er and Seah 2006; McDonald and Znati 2001) create arbitrarily large clusters. This is undesirable for our purposes (violates Requirements 1 and 4). Other multi-hop approaches (Ding, Holliday, and Celik 2005; Selvakennedy and Sinnappan 2007; Youssef et al. 2007; Spohn and Garcia-Luna-Aceves 2007; Dai and Wu 2005), mostly in the area of sensor networks, rely on static nodes and do not work in a mobile scenario (Requirement 2). Angione et al. (Angione et al. 2006) propose algorithms whose runtime increases with the number of nodes which is not feasible due to the number of participants we envision (Requirement 4). Other approaches (Lin and Chu 2000; Dai and Wu 2005) assume that connectivity between nodes does not change for a certain period of time (frozen period assumption) which is unrealistic and violates Requirement 2.

Liang and Haas (Liang and Haas 2000) propose a $r$-hop clustering algorithm for MANETs. However, they do not cope with fairness (Requirement 5) and the approach has a high maintenance overhead (Requirement 3). It involves periodic $2r$-hop broadcasts from each CH and a $r$-hop broadcast for every lost communication link. Similarly, in CONID (Chen et al. 2002) each node requires periodic update information about its $r$-hop neighborhood (in general $\Omega(r^2)$ messages per node). The Max-Min $r$-hop clustering algorithm (Amis et al. 2000) requires a complete periodic reclustering ($3r + 1$ messages per node). Our approach is much more efficient in term of message overhead since it tightly integrates with the public sensing system.

# 6 The Resource Manager

Our approach creates and maintains a $r$-hop cluster structure. Nodes are member of at most one cluster and have a distance of $\leq r$ hops to their CH. The CH coordinates cluster maintenance and periodically reports its *cluster hull*, i.e. the smallest polygon that covers all nodes within the cluster, to a resource manager component running in the infrastructure. Figure 1 shows our distributed query processing mechanism. First, an application issues a query to the resource manager running in the infrastructure. The manager forwards the query to all CHs (using cellular communication) whose cluster hulls overlap with the area-of-interest.

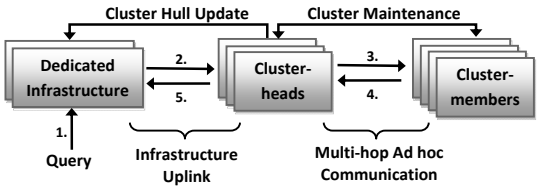


Figure 1: Distributed query processing

The CHs broadcast this query within their cluster using ad-hoc communication. Subsequently, the cluster members respond to the query and the CH collects the responses using a ring-based aggregation scheme. Finally, the CH sends the aggregated responses back to the infrastructure which then assembles the final query response.

The query broadcast and response collection mechanism within each cluster is also used to perform cluster maintenance: Some broadcasts contain an *advertisement* ($Adv$) that is sent to all $r$-hop neighbors. Hence, nodes can keep track of all CHs in a range of $r$ hops. If a node $n$ knows at least one CH besides its own, we say that $n$ is *covered*. In their response to an $Adv$, cluster members inform the CH about their available energy budget and whether they are covered. This way, the CH can infer if it is redundant (i.e. all its members are covered) or if there exists a better candidate for the role of a CH (i.e. a node with more available energy). Both, $Adv$ and $Adv$ responses, can be piggy-packed in application queries such that the maintenance overhead is negligible.

## 6.1 Cost Calculation

CHs use more energy than cluster members due to the use of cellular communication. Hence, CHs are elected based on their remaining energy budget to avoid draining individual nodes and to use the nodes' energy in a fair way. We express the suitability of a node $v$ to become a CH at time $t$ in with a cost function $c_v(t) \in [0, 1]$ (nodes with the lowest costs are the most suitable candidates).

Let $E^v_{rem}(t)$ be the remaining amount of energy of a node $v \in V$ at time $t$ and let $\delta^v_{rem}(t)$ be the user-specified time span for which the device should not be completely drained (see Section 2). Furthermore, let $P^v_{user}$ be the typical power drain of $v$ without public sensing, which can be determined based on past energy usage (e.g. moving average) of $v$. Hence, the expected available energy for public sensing in the time interval $[t, t + \delta^v_{rem}(t)]$ is $E^v_{PS}(t) = E^v_{rem}(t) - P^v_{user} \cdot \delta^v_{rem}(t)$. Nodes with a high ratio $P^v_{PS}(t) = E^v_{PS}(t)/\delta^v_{rem}(t)$ are the best CH candidates since they can contribute the most energy per time. We introduce constants $P_{min}$ and $P_{max}$ in order to normalize $P^v_{PS}(t)$ which is required for concepts we will discuss later. Values $P_{min} \leq P^v_{PS}(t) \leq P_{max}$ are linearly mapped to the interval $[0, 1]$ while values below/beyond the two thresholds $P_{min}, P_{max}$ are capped:

$$c_v(t) = \begin{cases} 0, \text{ if } P^v_{PS}(t) > P_{max} \\ 1, \text{ if } P^v_{PS}(t) < P_{min} \\ 1 - \frac{P^v_{PS}(t) - P_{min}}{P_{max} - P_{min}}, \text{ else.} \end{cases}$$

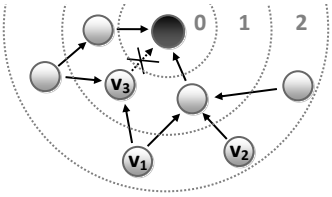


Figure 2: Aggregation in the ring topology.

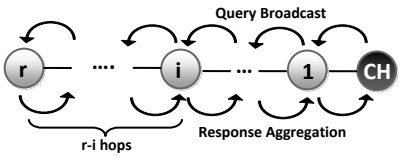


Figure 3: Query broadcast and response aggregation.

$P_{min}$ is the lower bound to the energy required by a CH (e.g. the power to maintain the ad-hoc interface). Conversely, $P_{max}$ is the value which allows a node to fulfill the role of CH safely. Both values can be derived either by experiments or based on energy models.

### 6.2 Aggregation and Query Model

Each query requests an aggregate of *resource states* of the cluster nodes. A resource state $R_v$ of a node $v$ contains information about $v$, e.g. its sensors, available memory, location, etc. An *aggregate* $A(R_{v_1}, R_{v_2}, \ldots)$ represents a number of such resource states. $A(R_{v_1}, R_{v_2}, \ldots)$ may be the result of an aggregation function (e.g. an average) or a simple list of the values. We assume that $A(.)$ is duplicate- and order-insensitive and can be merged with other aggregates. There are numerous data structures for this, supporting a broad range of aggregation functions such as SUM, MAX/MIN, AVG, and COUNT with sublinear space-complexity (Nath et al. 2004; Cormode and Muthukrishnan 2005).

### 6.3 Ring-based Data Aggregation

After receiving a query, a CH broadcasts it within its cluster (*broadcasting phase*). The subsequent aggregation of resource states (*aggregation phase*) is performed using a ring-based topology (Nath et al. 2004). In this scheme, each node is a member of a ring that is assigned initially, using a broadcast. The CH is the sole member of ring 0 and initiates this broadcast. A node is a member of ring $i$ if it receives the broadcast first from a node in ring $i-1$. Thus, a node's ring number $i$ equals the number of hops to the CH. The initial approach by Nath et al. assumes time-synchronized nodes. Each ring is assigned a time slot in which each member of the ring broadcasts (1-hop) its aggregate. This process starts with the outmost ring. Nodes in the next inner ring receive the aggregates, merge them, add their own resource state and broadcast the new aggregate, until the CH finally receives the cluster-wide aggregates. Figure 2 illustrates this: Node $v_1$ in Ring 2 broadcasts its resource state and two nodes in ring 1 receive it along with other resource states (i.e. that of node $v_2$). Even though the transmission of $v_3$ fails (e.g. due to collision), the resource state of $v_1$ arrives at the sink since it travels over multiple paths.

Due to its high robustness, we adopt this aggregation scheme. We adapt it for mobile scenarios and drop the assumption of time synchronicity. Figure 3 shows the query broadcast and aggregation phase of the new scheme along a single path in a cluster. Each node is denoted with its ring number ($r$ being the number of the outmost ring). The ring topology is built during the broadcasting phase for each request. Thus, the topology is updated immediately before it is actually used in order to cope with mobility. After the request broadcast has reached the nodes in ring $r$, the responses are propagated back to the CH. To drop the assumption of time synchronization, we propose the following scheme: After broadcasting a query, nodes in ring $i < r-1$ listen for subsequent request broadcasts of nodes in ring $i+1$. This way, they know their predecessors in the aggregation phase and can broadcast their own aggregate as soon as they have received all aggregates of their predecessors.

Assume a node $v$ receives a query-broadcast first from a node in ring $i-1$ at time $t$ (i.e. $v$'s ring is set to $i$). The time at which $v$ sends its query response depends on $i$:

- If $i = r$, $v$ sends its query response (only containing $v$'s resource state) immediately since $v$ has no predecessors.
- If $i < r-1$ and $v$ does not receive a query broadcast from a node in ring $i+1$ until time $t' = t + 2\Phi$ ($\Phi$ denotes the maximum message delay), $v$ knows that it has no predecessor in the aggregation phase (a predecessor would have forwarded the broadcast which $v$ would have overheard). Hence, $v$ can send its query response at time $t'$. If $v$ receives at least one broadcast before time $t'$, it sets a timeout $t_{sched} = t + 2\Phi(r-i)$ for sending its query response. This represents the maximum time required to broadcast the query for the remaining $r-i$ hops and propagating query responses back to $v$ (see Figure 3). $v$ keeps track of received broadcasts during that time. If $v$ has received the query responses of all its predecessors before $t_{sched}$, it sends its query response immediately.
- Finally, if $i = r-1$, $v$ will not hear a query broadcast from its predecessors since they are members of ring $r$ and do not need to forward it. However, as all ring-$r$ nodes will send their resource state immediately after receiving the broadcast, $v$ simply waits until time $t + 2\Phi$ before forwarding its aggregate.

CH ($r = 0$) sends its query response (containing the final aggregate) to the infrastructure instead of broadcasting it in the ad-hoc network.

### 6.4 Cluster Maintenance

The cluster maintenance uses the same aggregation mechanism discussed in Section 6.3. Each CH sends out an $Adv$ message (containing its ID), at least every $\Delta_{Adv}^{max}$ seconds, informing all nodes within $r$ hops of its existence. All nodes that receive an $Adv$ store it in a list and delete entries that are older than $\Delta_{Adv}^{max}$. Upon receipt of an $Adv$, a node may join the new cluster. This happens in two cases:

1. A node has no assigned cluster.
2. A node is already assigned to a cluster but the $Adv$ traveled over less hops than the last $Adv$ of its current cluster.

The latter ensures that nodes are always connected to the closest CH, balancing the size of clusters. Joining simply means that a node changes its current Cluster-ID to the ID of the new CH and provides answers to future queries/$Adv$s that are addressed to the new cluster.

If a node $v$ joins the system or if it did not receive an $Adv$ from its CH for a time $2 \cdot \Delta_{Adv}^{max}$, it switches into the *unclustered* state. After remaining in this state for time $c_v(t) \cdot \Delta_{Adv}^{max}$, $v$ declares itself as CH and sends an $Adv$. However, if it receives an $Adv$ from another CH $c$ during this time, it joins $c$. Waiting for a time which is proportional to the cost ensures that only nodes with the highest energy budgets become CHs. Furthermore, it prevents that many nearby nodes declare themselves a CH concurrently, e.g. when a CH moves away, switches off or fails.

$Adv$s and as their answers can be piggybacked on queries and responses, respectively. If a query is about to be broadcast by a CH and the last $Adv$ was sent more than $\Delta_{Adv}^{min}$ seconds ago, an $Adv$ is attached to the query. The difference between an $Adv$ and a query broadcast is that the former is broadcast along $r$ hops while the latter is only forwarded within the corresponding cluster. Hence, nodes of ring $i < r$ that are not members of the corresponding cluster must forward $Adv$s proactively. However, in our evaluation, we show that only very few proactive maintenance messages are required. Defining a suitable $\Delta_{Adv}^{min}$ keeps this overhead low even if the query rate is very high. Like query broadcasts, $Adv$s are followed by an $Adv$ *response* ($Adv\_R$) which is aggregated using our ring-based scheme. Since only members of the corresponding clusters provide an answer for queries and $Adv$s, there is no message overhead for cluster maintenance if piggybacking can be used. Only if no $Adv$ has been sent (piggybacked or not) for $\Delta_{Adv}^{max}$ seconds, it is issued proactively by the CH.

An $Adv\_R$ contains three aggregates of small constant size which may trigger different reactions by the CH node $v_{CH}$ that has sent the $Adv$:

1. **Redundant CH.** If $v_{CH}$ knows that all its members are covered, $v_{CH}$ is redundant and can be removed. This only requires sending a single boolean aggregate $A_R$ per node $v$ in ring $i$ with $A_R = true$ iff $v$ is covered *and* all nodes that are members of ring $i+1$ in range of $v$ reported $A_R = true$. If only a single node is not covered we have $A_R = false$.

   If $v_{CH}$ receives a $A_R = true$, it concludes that it is redundant and broadcasts a final $r$-hop *dissolve* ($Dis$) message. Any node that receives a $Dis$, removes $v_{CH}$ from its CH list and, if it is a member of the cluster, switches to the closest CH in its advertisement-list.

2. **Fairness.** To achieve a fair distribution of energy consumption, the CH role is changed frequently: Each $Adv\_R$ includes the ID of the node $v_{min}$ with the lowest cost. This is implemented using an aggregate $A_C$ storing the ID and the cost of $v_{min}$. $A_C$ is propagated through $Adv\_R$ messages and updated along the way analogously to $A_R$ until it arrives at the CH. If $c_{v_{min}} + \Theta < c_{v_{CH}}$ ($\Theta \in [0, 1]$ is called the *cost threshold*) $v_{CH}$ sends a $r$-hop $Dis$ containing the identifier of $v_{min}$. As soon as $v_{min}$ receives this $Dis$ and finds out that it is the member with the lowest cost, it becomes CH immediately and sends an $Adv$. Note that there may be some nodes that are not covered by $v_{min}$. However, they will switch to the unclustered state and subsequently join other clusters or become CHs.

3. **Cluster Hull.** Each $Adv\_R$ also includes the (preliminary) cluster hull $A_H$ covering all nodes whose resource states it contains. Assuming $A_H$ is represented as a polygon with fixed amount of edges (e.g. bounding box), $A_H$ can be implemented as a constant-sized aggregate. At regular intervals, $v_{CH}$ sends $A_H$ to the infrastructure such that queries can be directed to the right CHs.

$A_R$, $A_C$, and $A_H$ are small and of constant size. Thus, they can be piggybacked along query broadcasts and responses with negligible overhead.

# 7 Evaluation

We implemented the resource manager using the OMNet++ simulator (Varga ) with the INETMANET extension. We simulated cellular communication using an energy model for GPRS (Balasubramanian, Balasubramanian, and Venkataramani 2009), and ad-hoc communication using IEEE802.11g interfaces at 54Mbit based on a model for a HTC G1 phone (Xiao et al. 2010). We assume that the cellular interface is turned on anyway (i.e. maintenance energy is not counted), in contrast to the WiFi interfaces which must be switched on due to our resource manager (i.e. we count the maintenance energy for it). This represents worst-case conditions for our evaluation. Under other conditions, the gain produced by our approach would be even higher.

For ad-hoc communication, the widely-used *Two-Ray Ground reception model* was used to determine path-loss with a maximum communication distance of $100m$. We used the graph-based mobility model (Tian et al. 2002) with a street-graph of the city-center of Stuttgart that covers an area of $1km^2$. Nodes move with a speed uniformly selected between 0 and $3m/s$ (i.e. pedestrian speed). We executed 10 simulations per parameter-setting, each for $1800s$.

We used the following parameter settings: $\Delta_{Adv}^{max} = 3s$, $\Delta_{Adv}^{min} = 1s$, $\Phi = 90ms$. We simulated 700 nodes, each node $v_i$ providing a resource state of size $|R_{v_i}| = 32$ bytes. Let $S_A(n) \in \mathbb{N}$ denote the space required to represent an aggregate $A$ which contains a total of $n$ resource states. We chose generic aggregation functions (per query) selected uniform randomly among three different typical representatives: ($A_1$) constant aggregation (e.g. MIN/MAX), i.e. $S_{A_1}(n) = |R_{v_i}|$, ($A_2$) logarithmic aggregation (e.g. sketches (Cormode and Muthukrishnan 2005)), i.e. $S_{A_2}(n) = |R_{v_i}|(1 + log(n))$, ($A_3$) linear aggregation (list of filtered resource states), i.e. $S_{A_3}(n) = |R_{v_i}| \cdot \alpha \cdot n$ with selectivity $\alpha \in [0, 0.5]$. Queries of size 128 bytes were sent Poisson-distributed with a mean query rate of 1 query/s to all CHs. CHs reported their current cluster hull to the

infrastructure after each executed query. We set $P_{min} = 650mW$ (ad-hoc maintenance power) and $P_{max} = 4W$.

### 7.1 Results

We compare our approach with an approach where *each* MN receives and responds to application queries using cellular communication only (i.e. with Wifi interface turned off). We call this approach *Pure IS*, which is essentially a special case of our approach with $r = 0$, i.e. every node is a CH.

**Energy** In Figure 4, we show the average drained energy per node vs. the increasing node density. Our approach saves up to $65\%$ of energy. Both, increasing the cluster-radius and the density, decreases the amount of drained energy since clusters get larger (in terms of members) and less nodes use expensive cellular communication. The additional energy savings get smaller as we increase the radius/density since the drained energy converges to that required for WiFi. If we decrease the density, the energy usage converges towards that of the Pure IS approach since at some point as each node is isolated and creates a cluster for itself.

As we increase the query rate (Figure 5), we observe significant energy savings for rates above $\approx 0.25$ queries/s. GPRS keeps the radio turned on for a certain time after sending a packet to reduce the delay for the following packets. Hence, above a certain query rate, it is always on which explains the steep rise and the saturation effect for $> 1$ queries/s.

**Infrastructure Load** The data volume per MN that eventually reaches the infrastructure (Figure 6) behaves qualitatively similar to the drained energy (Figure 4). The amount of data is inversely correlated to the size of clusters due to data aggregation. Our approach reduces the amount of data that has to be processed in the infrastructure by up to $90\%$.

**Query Execution Time** Figure 7 shows that the average time between the first query broadcast by the CH and sending the final aggregate to the infrastructure is much smaller than for the original ring-based algorithm by Nath et al. (Nath et al. 2004).

**Clustering Maintenance Overhead** Figure 8 shows the average number of cluster maintenance messages per node per maximum cluster update period (i.e. $\Delta_{Adv}^{max}$). The overhead increases with $r$ since a larger $r$ implies that more proactive (non-piggybacked) messages are necessary. Increasing the query rate leads to more piggybacking opportunities and, thus, the overhead decreases to $0.5$ queries/s. For larger query rates, the overhead increases slightly since more $Adv$s are issued. However, due to the minimum time between advertisements ($\Delta_{Adv}^{min}$), this value converges. Due to piggybacking, the overhead is well below existing clustering approaches which require more than $2r$ messages per node and per update interval (see Section 5).

**Fairness** To evaluate the concept of fair energy consumption, we ran simulations over $10000s$ and chose the initial energy budget per node uniform randomly from $[7500J, (7500 + n)J]$ for different $n$. The WiFi maintenance energy over the simulation duration is $6500J$. For small $n$, nodes need nearly all available energy just for being cluster member and can be CH only for a small fraction of time. Hence, the total amount of energy in the system is just above the minimum.

Figure 9 shows the number of devices $d$ (out of 700) whose battery was drained during the simulation against the energy spread $n$ and cost threshold $\Theta$. $d$ decreases as the $\Theta$ decreases since CHs change *more aggressively* if they have more cost than some member. $d$ decreases as the overall amount of energy in the system (controlled by $n$) increases. For $n \geq 2000$, a small enough $\Theta$ guarantees that no device is drained, even if the total energy in our system is only slightly above the absolute minimum. For $n = 1000$, a small fraction of batteries ($< 10\%$) are drained even for a low cost threshold of $\Theta = 0.1$. This is due to the fact that some nodes *have* to play the CH role in some situations, i.e. when they are either isolated or surrounded only by nodes with low energy. Furthermore, even for $\Theta = 1$, our system maintains a certain fairness since CHs change over time due to mobility and new CHs are preferably those nodes with the highest energy budget.

**Query Delivery Ratio** In Figure 10, we vary the node density: For low densities, decreasing the cluster-radius leads to a slightly better delivery ratio since we have a sparse communication graph which can be easily disrupted (due to mobility) during the execution of a query. Generally, the probability that a given resource state reaches the CH over a specific path decreases with the path's length. However, by increasing the node density, we increase the number of paths each resource state travels. Hence, for densities larger than 700, the longer path length (larger $r$) is compensated by the additional redundancy. For increasing node density, the delivery ratio decreases because the number of collisions increases. However, even for a very high density of 2000 $nodes/km^2$, the delivery ratio still exceeds $93\%$.

Figure 11 shows that increasing the query rate gradually decreases the delivery ratio after $\approx 0.5$ queries/s since multiple queries overlap and thus create additional collisions. However, the delivery ratio degrades slowly and can be countered by batching queries.

Figure 12 shows the effect of an increasing resource state size on the delivery ratio. For constant space complexity, increasing the resource state does not have affect the delivery ratio since the small size of the aggregate leads to a small probability of collisions. For linear aggregation, the size of the aggregate (and thus the collision probability) increases proportionally to the number of included resource states, leading to a faster drop in the delivery ratio. However, even for large resource states, the delivery ratio stays above $93\%$.

## 8 Conclusions and Outlook

We investigated resource management in public sensing. Based on a system model of mobile sensor devices and a hybrid communication model, we proposed a cluster-based approach that exploits ad-hoc communication and tightly integrates cluster maintenance with query processing in order to minimize energy consumption. We have shown that our approach saves up to $65\%$ energy and decreases the load on

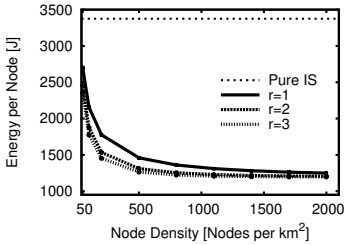


Figure 4: Energy vs. density

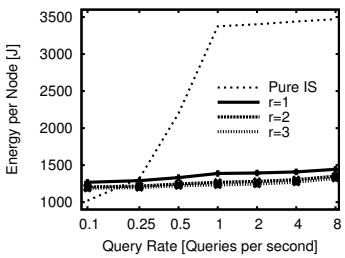


Figure 5: Energy vs. query rate

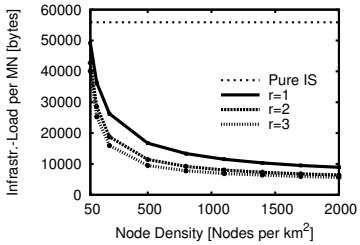


Figure 6: Infr. load vs. density

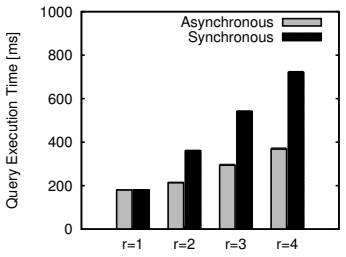


Figure 7: Query execution time

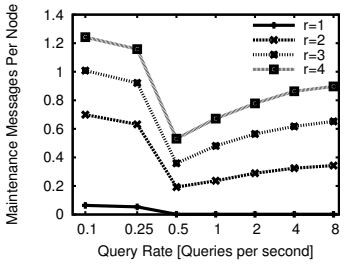


Figure 8: Overhead vs. query rate

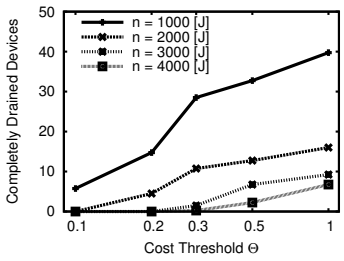


Figure 9: Drained devices

the infrastructure by up to $90\%$. These numbers are attained while a high average delivery ratio of $93\%$ is achieved. The clustered approach together with the reduction of infrastructure load leads to a very good scalability – a property that is vital for public sensing systems.

The resulting resource management system is a crucial enabling technology that allows for large-scale and highly flexible sensing applications on top of a generic public sensing substrate. This is an important step forward from today's state of the art towards realizing the general vision of public sensing.

In future work, we will investigate query processing within the infrastructure more closely and develop a distributed approach that minimizes communication costs and utilizes caching.

## References


Abbasi, A. A., and Younis, M. 2007. A survey on clustering algorithms for wireless sensor networks. *Comput. Commun.* 30:2826–2841.

Amis, A. D.; Prakash, R.; Vuong, T. H.; Huynh, D. T.; P, T. H.; Dung, V.; and Huynh, T. 2000. Max-min d-cluster formation in wireless ad hoc networks. In *in Proceedings of IEEE INFOCOM*, 32–41.

Angione, G.; Bellavista, P.; Corradi, A.; and Magistretti, E. 2006. A k-hop clustering protocol for dense mobile ad-hoc networks. In *ICDCSW '06: Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems*.

Badonnel, R.; State, R.; and Festor, O. 2005. Management of mobile ad hoc networks: information model and probe-based architecture. *Int. J. Netw. Manag.* 15(5):335–347.

Balasubramanian, N.; Balasubramanian, A.; and Venkataramani, A. 2009. Energy consumption in mobile phones: a measurement study and implications for network applications. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*.

Campbell, A. T.; Eisenman, S. B.; Lane, N. D.; Miluzzo, E.; Peterson, R. A.; Lu, H.; Zheng, X.; Musolesi, M.; Fodor, K.; and Ahn, G.-S. 2008. The rise of people-centric sensing. *IEEE Internet Computing* 12(4):12–21.

Chen, G.; Nocetti, F.; Gonzalez, J.; and Stojmenovic, I. 2002. Connectivity based k-hop clustering in wireless networks. 2450 – 2459.

chung Shen, C.; Jaikaeo, C.; Srisathapornphat, C.; and Huang, Z. 2003. The guerrilla management architecture for ad hoc networks. *IEEE Communications Magazine* 41:108–115.

Cormode, G., and Muthukrishnan, S. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms* 55(1):58–75.

Cornelius, C.; Kapadia, A.; Kotz, D.; Peebles, D.; Shin, M.; and Triandopoulos, N. 2008. Anonysense: privacy-aware people-centric sensing. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*.

Cuff, D.; Hansen, M.; and Kang, J. 2008. Urban sensing: out of the woods. *Commun. ACM* 51(3):24–33.

Dai, F., and Wu, J. 2005. On constructing k-connected k-dominating set in wireless networks. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, 81.1. Washington, DC, USA: IEEE Computer Society.

Ding, P.; Holliday, J.; and Celik, A. 2005. Distributed

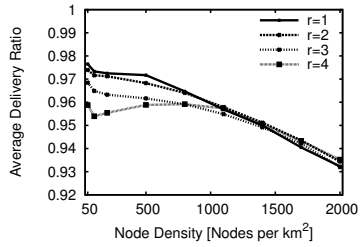


Figure 10: Delivery ratio vs. density

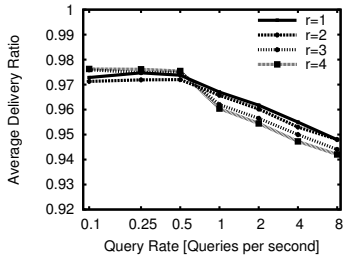


Figure 11: Delivery ratio vs. rate

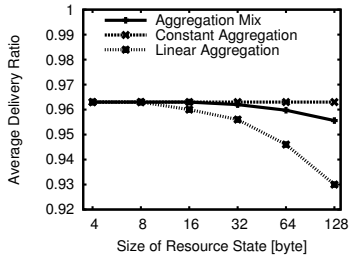


Figure 12: Delivery ratio vs. size


energy-efficient hierarchical clustering for wireless sensor networks. *Distributed Computing in Sensor Systems* 322–339.

Er, I. I., and Seah, W. K. 2006. Performance analysis of mobility-based d-hop (mobdhop) clustering algorithm for mobile ad hoc networks. *Computer Networks* 50:3375 – 3399.

Kanjo, E.; Bacon, J.; Roberts, D.; and Landshoff, P. 2009. Mobsens: Making smart phones smarter. *Pervasive Computing, IEEE* 8(4):50–57.

Kazemi, H.; Hadjichristofi, G.; and DaSilva, L. A. 2008. Mman - a monitor for mobile ad hoc networks: design, implementation, and experimental evaluation. In *WiNTECH '08: Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, 57–64. New York, NY, USA: ACM.

Lane, N. D.; Eisenman, S. B.; Musolesi, M.; Miluzzo, E.; and Campbell, A. T. 2008. Urban sensing systems: opportunistic or participatory? In *HotMobile '08: Proceedings of the 9th workshop on Mobile computing systems and applications*.

Liang, B., and Haas, Z. J. 2000. Virtual backbone generation and maintenance in ad hoc network mobility management. In *INFOCOM*, 1293–1302.

Lin, H.-C., and Chu, Y.-H. 2000. A clustering technique for large multihop mobile wireless networks. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 2, 1545 –1549 vol.2.

Lochert, C.; Scheuermann, B.; and Mauve, M. 2010. A probabilistic method for cooperative hierarchical aggregation of data in vanets. *Ad Hoc Networks* In Press, Uncorrected Proof:–.

Lu, H.; Lane, N. D.; Eisenman, S. B.; and Campbell, A. T. 2010. Bubble-sensing: Binding sensing tasks to the physical world. *Pervasive and Mobile Computing* 6(1):58 – 71.

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

McDonald, B., and Znati, T. F. 2001. Design and performance of a distributed dynamic clustering algorithm for ad-hoc networks. In *SS '01: Proceedings of the 34th Annual Simulation Symposium (SS01)*, 27. Washington, DC, USA: IEEE Computer Society.

Nath, S.; Gibbons, P. B.; Seshan, S.; and Anderson, Z. R. 2004. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 250–262. New York, NY, USA: ACM.

Parker, A.; Reddy, S.; Schmid, T.; Chang, K.; Saurabh, G.; Srivastava, M.; Hansen, M.; Burke, J.; Estrin, D.; Allman, M.; and Paxson, V. 2006. Network system challenges in selective sharing and verification for personal social, and urban-scale sensing applications. In *In HotNets*.

Ramach, K. N.; Belding-royer, E. M.; and Almeroth, K. C. 2004. Damon: A distributed architecture for monitoring multi-hop mobile networks. In *In Proceedings of IEEE SECON*.

Selvakennedy, S., and Sinnappan, S. 2007. An adaptive data dissemination strategy for wireless sensor networks. *Int. J. Distrib. Sen. Netw.* 3(1):23–40.

Spohn, M. A., and Garcia-Luna-Aceves, J. J. 2007. Bounded-distance multi-clusterhead formation in wireless ad hoc networks. *Ad Hoc Netw.* 5(4):504–530.

Sun, G.; Chen, J.; Guo, W.; and Liu, K. 2005. Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs. *Signal Processing Magazine, IEEE* 22(4):12 – 23.

Tian, J.; Hahner, J.; Becker, C.; Stepanov, I.; and Rothermel, K. 2002. Graph-based mobility model for mobile ad hoc network simulation. In *35th Annual Simulation Symposium*, 337 – 344.

Tuduce, C., and Gross, T. 2004. Resource monitoring issues in ad hoc networks. 259 – 264.

Varga, A. The OMNet++ Simulator. http://www.omnetpp.org/.

Xiao, Y.; Savolainen, P.; Karppanen, A.; Siekkinen, M.; and Ylä-Jääski, A. 2010. Practical power modeling of data transmission over 802.11g for wireless applications. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*.

Youssef, A.; Younis, M. F.; Youssef, M.; and Agrawala, A. 2007. Establishing overlapped multihop clusters in wireless sensor networks. *Int. J. Sen. Netw.* 2(1/2):108–117.