Service-Oriented Middleware for Large-Scale Mobile Participatory Sensing

Sara Hachem, Animesh Pathak, Valérie Issarny

Service-Oriented Middleware for Large-Scale Mobile Participatory Sensing

Sara Hachem, Animesh Pathak, Valerie Issarny

Inria Paris-Rocquencourt France
{sara.hachem, animesh.pathak, valerie.issarny}@inria.fr

Abstract

In this paper, we introduce MobIoT, a service-oriented middleware that enables large-scale mobile participatory sensing. Scalability is achieved by limiting the participation of redundant sensing devices. Precisely, MobIoT allows a new device to register its services only if it increases the sensing coverage of a physical attribute, along its expected path, for the set of registered devices. We present the design and implementation of MobIoT, which mobile devices use to determine their registration decision and become accessible for their services. Through experiments performed on real datasets, we show that our solution scales, while meeting sensing coverage requirements.

1. Introduction

In its traditional definition, *Participatory Sensing*, also known as mobile crowdsensing [1] assigns mobile devices to form participatory sensor networks that enable public and professional users to gather, analyze and share local knowledge [2].

In existing participatory sensing solutions, it is common to adopt a decoupled approach where the sensing of and the querying for information operations are *asynchronous* (as seen in solutions presented in [3]), i.e., the sensor is tasked to sample the real world periodically, continuously or based on some event, etc., but is not triggered by a user's request. In many cases, the latter leads to more appropriate results. Indeed, sensing the world independently of the measurement request time and intended use of the data is not necessarily the best option; such approach may lead to large out-dated streams of data that can be of low benefit while more useful up-to-date measurements are missing. The issue of data freshness versus response time arises in this case. Our position is that many scenarios can handle a slower response as long as data is more up to date. Further, most existing solutions in the participatory sensing field are domain-specific, designed to monitor either the environment, traffic, health, or social interactions, etc. [3]. However, with the increasing popularity of participatory sensing, providing common support to different sensing applications belonging to different categories becomes a must, as it saves development efforts and time.

To that end, our work focuses on domain-agnostic support for large-scale mobile participatory sensing. When performing large-scale sensing —where sensing devices

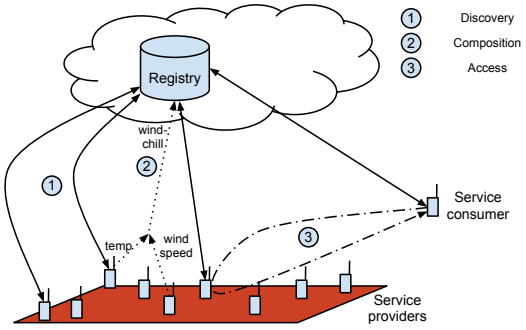

Figure 1: The operations of a service-oriented mobile participatory sensing system.

are not necessarily in the geographical vicinity of the requestors— adopting a distributed approach does not perform well since routing and communication costs will grow unmanageable when millions of devices become involved. Therefore, the information about devices willing to provide their sensing measurements at the time of the query must be made available somewhere, e.g., in a registry. Hence, we augment the definition of participatory sensing with the identification of two main components:

1. **Mobile Devices**: Computers, smartphones, tablets, etc., hosting different sensors and actuators. Those devices are numerous and have limited energy resources.
2. **Registry**: Web-based component that holds metadata of sensing services. The component has performance capacity directly proportional to monetary cost.

The two components interact with each other in two phases (shown in Figure 1):

1. **Registration**: Each mobile device is expected to periodically advertise its services to the Registry which holds their metadata.
2. **Lookup and Access**: When data about the physical environment of a remote region is required, the Registry is queried for devices with the relevant sensing services to then access them and acquire their measurements, which can be processed to obtain the desired results.

For example, consider a query of the form "what is the average crowd level at the *Jardin de Tuileries* in Paris right now?". The answer can be generated by sensing the sound level surrounding each mobile device at the location of interest and using it to compute the crowd level through some mathematical expression. Although the solution above seems straightforward, there are several important challenges to address (for a detailed discussion, we refer readers to [4]). The first issue to overcome is the *heterogeneity* of devices as participatory sensing networks are bound to contain devices from an assortment of vendors, with varying sensing characteristics such as error distributions, sampling rates, spatial resolution, and so on. Second, the network is characterized by an *unknown and dynamic structure* resulting from the mobility of devices [5]. Last but not least, any approach must be *scalable* as mobile participatory sensing involves millions of mobile devices, a number that has important repercussions on communication, storage, memory and time consumption costs.

To address the above challenges, we adopt the service-oriented paradigm [6, 7] to decouple the high level sensing system logic from the low level network heterogeneity and mobility issues. We provide a service-oriented middleware that supports the following, traditional functionalities: *Discovery* of, *Composition* of, and *Access* to *sensing services* (Figure 1). Sensing services are, in this context, software logic that abstracts sensors and have functional (e.g., the attribute to measure) and non functional (e.g., measurement accuracy) properties. *Discovery* is used by devices wishing to provide measurements of the real world, to publish (register) their sensing services in registries that hold service metadata and to look for sensing services that can satisfy a sensing request. *Composition* of services is used when discovered sensing services are unable to fulfill the request. In such case, other registered sensing services are combined to provide a new, more convenient functionality. The composed services can further be used for more complex compositions, all of which should be defined, then executed in a composition engine provided by the middleware. Finally, *Access* enables interaction with discovered services.

Traditional approaches in mobile participatory sensing require input from as many sensors as possible to have the most accurate information about the real world [8]. We postulate that this should not be the governing rule for two main reasons. First, given today's abundant sensor availability through smart mobile devices, involving them all to partake in a sensing task provides redundant measurements that do not necessarily benefit the sensing quality noticeably, yet increase the communication and energy consumption costs. Second, in many cases, having the most accurate answer is not always the priority. In some situations, users might be satisfied with less accurate answers that have a "good enough" geographical sensing coverage of the real world. Consequently, this allows us to address the data freshness versus response time issue, mentioned earlier, that on-demand sensing may face. That is because sensing tasks triggered by users' requests can now provide up-to-date data with lower measurement latency as less devices are involved, and therefore less communication and computation efforts must be paid. Our contribution builds on that logic as we design our middleware to limit the number of participating sensing devices, based on the devices' mobility characteristics, while satisfying required sensing coverage through a revisited version of service discovery, namely in registration. Sensing coverage refers to the geographical area covered, i.e., sensed by a sensing device.

We first provide an approach that determines, with high accuracy, if the path of a new device will intersect with those of registered devices by acquiring information on their mobility. It is then possible to determine, from the computed result, if a certain percentage of the new device's path can already by covered, i.e., sensed, by the registered ones. If so, the new device is considered redundant and need not register. However, this approach is costly and requires a computation time that increases linearly with the number of registered devices. To enhance the performance of our solution, we substitute the mobility paths information with a mobility model that allows us to estimate the mobility of registered devices and then compute a probability of the possible coverage (introduced in our previous work in [9]). Although this probabilistic approach is less accurate, it is faster and still meets required sensing coverage.

By adopting probabilistic device registration, the overall sensing system no longer requires expensive registries with high processing capabilities and can instead use

cheaper ones with basic capacities. Furthermore, reducing the number of participating devices benefits the lookup and access phases since having less devices to search through and access accelerates the query answering process. Second, having only devices that increase the sensing coverage register their services eliminates the system's need to deal with large amounts of redundant data. Third, the fewer devices get involved, the more energy is saved until the —hitherto redundant— devices are needed.

Mobile coverage estimation has been used before for resource allocations [10], routing techniques [11, 12], or for robotic entities with controlled mobility [13]. The novelty of our approach is that it adopts mobile coverage estimation to provide "good" sensing coverage while having less mobile devices register their services. Our contributions in this paper are as follows:

- The design and analysis of a deterministic registration approach for mobile devices that computes a registration decision based on accurate knowledge of mobility paths of already registered nodes (Section 2).
- The mathematical modeling, design and analysis of a probabilistic registration approach for mobile devices that computes a registration decision based on an estimation of the mobility paths of already registered nodes (Section 3).
- The design and implementation of MobIoT, a domain-agnostic service-oriented middleware that builds on the proposed registration approach to enable scalable and mobile participatory sensing (Section 4).
- The evaluation of the performance of our probabilistic registration approach with real mobility traces and its comparison to the deterministic approach (Section 5).

Finally, we present the related works in Section 6 and the conclusion and future work in Section 7.

2. Deterministic Registration Approach

The problem we solve is twofold: Providing **domain-agnostic support for on demand** participatory sensing while guaranteeing the **scalability** of the solution by controlling whether or not devices should register their services.

2.1. System Model

The mobile sensor network comprises one or several types of sensors hosted on mobile devices spread over a total area of interest A. The registration decision for a new mobile device entering the network is based on the device's path and whether or not this path will be covered by other known mobile devices, i.e., whether other devices with similar sensing capacities will be present at this device's future locations when it passes through them. The path is represented as a sequence of (x, y) coordinates and timestamps representing the times at which the device will reach each pair of coordinates. Notations are summarized in Table 1. Descriptions of the network and device characteristics on which we build our solution follow.

Symbol	Meaning
	Mobility
D	Diffusion constant
v	Speed of mobile devices
	Deployment/Area
A	Total area of deployment
C_{l,r_τ}	Circle of center l and radius r_τ
Σ_{l,r_τ}	Smallest square outside C_{l,r_τ} with edges parallel to the coordinate axes
σ_{l,r_τ}	Largest square inside circle C_{l_i,r_τ} with edges parallel to the coordinate axes
	Device
T	The set of all sensor types
R	The set of all registered devices
k	A registered (mobile) device $\in R$
τ	A type of sensor, $\tau \in T$
R_τ	The subset of R with devices hosting sensors of type τ
S_τ	Sensor of type τ hosted on a device $\in R_\tau$
e_τ^j	Expansion set $e_\tau^j \subseteq T$ containing sensor types that together can substitute a sensor of type τ
E_τ	Set of all expansion sets that can each replace a sensor of type τ
r_τ	The identical sensing range r_τ of devices hosting sensors of type τ
	Registration
t_0	Time at which the new device joins the network
l_i	Location of the new device at time t_i
l_0^k	Location of device k at time t_0
l_i^k	Location of device k at time t_i
δ_i^k	Displacement of device k during $t_0 \rightarrow t_i$
d_{max}^i	Maximum distance from beyond which a device can not reach l_i at time t_i

Table 1: Notations used in this paper.

Network Characteristics. The mobile network has the following characteristics:

- **Large number of sensors**: The network consists of a large set R of registered mobile devices with each embedding several types of sensors. R_τ is a subset of R containing devices hosting sensors of type τ, and T is the set of possible types of sensors. A sensor is denoted as $S_\tau,\ \text{where}\ \tau \in T$.
- **Dynamic structure**: The most commonly used devices in mobile participatory sensing are smart phones hosting several types of sensors. Each mobile device κ moves to different locations l_i^κ, starting from l_0^κ.

Mobile Device Characteristics. A mobile device has the following characteristics:

- **Mobility**: All devices in the network are mobile. Devices following the same mobility model (pedestrian, in vehicles, etc.) have, on average, a constant speed v (an assumption commonly made in existing solutions such as [14]).
- **Location-awareness**: Each device is equipped with the ability to identify its location as (x, y) coordinates (e.g., GPS receiver) [5].
- **Path awareness**: Each device in our system is aware of the path it will follow and the final destination to reach during a mobility period. It is increasingly common

for individuals using vehicles, even pedestrians, to follow paths specified by the navigation systems on their devices.[1] If users do not have navigation systems on their devices, it is possible for them to provide their future locations as input or estimate their most likely future path using some prediction techniques.

- **Fixed, identical sensing range**: All sensors of the same type τ in the network have an identical sensing range r_τ, and each sensor has a 360° coverage and can sense events in a circle C_{l_i,r_τ} with radius r_τ and centered at location l_i. This is known as the boolean disk model [15].

2.2. Deterministic Registration Solution

For the sensing system to be truly scalable, it must be able to support a large number of devices (millions and more) spread over large areas. To that end, our goal is to limit the number of registered devices while meeting minimum sensing coverage requirements. We present an approach to reach this goal through a registration approach that determines whether or not registered devices will cover the path of the new mobile device. If the coverage of already registered devices is above the minimum required coverage, the new device does not need to register its sensing service and vice versa. The minimum required coverage (threshold) is sensor-specific and specified in a *knowledge base* describing metadata about sensors, actuators, physics, etc. (for more details we refer the reader to [16]). We take two cases into account when computing coverage:

1. *Direct coverage*: only the set of devices hosting a service of similar type to the new device's service is taken into account when computing coverage.
2. *Coverage by composition*: all possible service compositions are computed through an expansion process where types of sensors that together substitute a sensor S_τ are determined. Services abstracting such sensors are *expansion services*. Each possible combination of expansion services is referred to as an expansion set e_τ^j and $E_\tau = \cup_j e_\tau^j$ is the set of all expansion sets for a sensor type τ. This information is modeled in the knowledge base mentioned above. For example, a wind-chill sensor (type τ) can be expanded into (substituted by) a thermometer (type τ_1) and an anemometer (type τ_2). In such case, $e_\tau^j = \{\tau_1, \tau_2\}$.

To compute coverage, the approach requires details on the mobility paths of devices. Specifically, upon actual registration, each device is required to send the path it will follow to the registry. The paths of registered devices that provide a similar sensing service to the one hosted on the new device will be checked to determine if any of them will intersect with the new device. For this purpose, the registry estimates the location of the devices based on their paths at the exact time t_i at which the new device is supposed to cross location l_i on its path. The estimation is done through an interpolation function that finds the location of a registered device κ at time t_j^κ and t_{j+1}^κ such that $t_j^\kappa \leq t_i < t_{(j+1)}^\kappa$ and estimates its location at time t_i. The deterministic registration decision algorithm is presented in Algorithm 1, which uses methods presented in Algorithm 2 and Algorithm 3.

[1] Services such as `http://www.google.com/mobile/maps/` provide turn-by-turn driving and walking navigation.

Algorithm 1 Deterministic Registration decision process for service of type τ

Input: $L, r_\tau, \tau, threshold, R, E_\tau$
Output: $decision \in \{true, false\}$
let $C \leftarrow \emptyset$ // C is the set of covered location
for $each\ l_i \in L$ **do**
  $intersect \leftarrow false$
  for $each\ s \in R_\tau$ **do**
    $L^s \leftarrow s.path$
    // Check if s is at l_i at time t_i.
    if $intersects(l_i, t_i, L^s, r_\tau)$ **then**
      $intersect \leftarrow true$
      $C \leftarrow C \cup \{l_i\}$
      $break$ // Break if intersection found and check next location
    end if
  end for
  if $intersect = false$ **then**
    for $each\ e_\tau^j \in E_\tau$ **do**
      $intersect \leftarrow false$
      for $each\ type \in e_\tau^j$ **do**
        for $each\ ss \in R_{type}$ **do**
          $L^{ss} \leftarrow ss.path$
          if $intersects(l_i, t_i, L^{ss}, r_{type})$ **then**
            $intersect \leftarrow true$
            $break$
          else
            $intersect \leftarrow false$
          end if
        end for
      end for
      if $intersect = true$ **then**
        $C \leftarrow C \cup \{l_i\}$
        $break$
      end if
    end for
  end if
end for
$c \leftarrow getCoveragePercentage(L, C, r_\tau)$
if $c < threshold$ **then**
  $decision \leftarrow true$
else
  $decision \leftarrow false$
end if

Complexity Analysis: The algorithm terminates when a decision is reached after the end of the outermost loop. All the elements in the loops (both inner and outer) are finite and therefore the loops are sure to terminate. In the worst case scenario, the algorithm will check all registered services in R, all expansion sets in E_τ and expansion types in each expansion set e_τ^j leading to a time complexity O $\left(|L||R||E_\tau||e_\tau^j|\right)$. $|E_\tau|$, $|e_\tau^j|$ and $|L|$ are constant and can be ignored, leaving the time complexity to be O $(|R|)$. In the best case scenario the algorithm will find intersections with the first registered service with no expansions, the best case time complexity is therefore $\Theta(1)$. Since we do not perform any preprocessing, and we do not use any complex data structures, the space complexity of the deterministic registration algorithm is O $(|L||R|)$; the set of covered locations C will have $|L|$ locations at most if all locations on the new device's path will be covered by registered devices.

Our registration approach can be regarded as a search problem since we are looking for intersections between the paths of registered devices and that of the new device. In traditional search solutions, a preprocessing step is preformed to optimize the search process (e.g., sorting data prior to executing the search requests). In our algorithm the data to sort is the locations of registered devices at the time at which a new device joins the network, which in the traditional case, would have a time complexity

Algorithm 2 $intersects$: to compute whether or not a device crosses l_i at time t_i

Input: l_i, t_i, L^s, r_τ
Output: $intersect : true, false$
for $l_i^s \in L^s$ **do**
  if $t_j^s \leq t_i < t_{j+1}^s$ **then**
    $l_i^s \leftarrow l_j^s + (l_{j+1}^s - l_j^s) * \frac{t_i - t_j^s}{t_{j+1} - t_j^s}$
    if $(l_i^s.x - l_i.x)^2 + (l_i^s.y - l_i.y)^2 \leq r_\tau^2$ **then**
      $intersect \leftarrow true$
      $break$
    else
      $intersect \leftarrow false$
    end if
  end if
end for

Algorithm 3 $getCoveragePercentage$: To compute the percentage of coverage of the device's path

Input: L, C, r_τ
Output: p
$l_{old}.x \leftarrow l_0.x, l_{old}.y \leftarrow l_0.y, x_{old} \leftarrow 0, y_{old} \leftarrow 0, td \leftarrow 0, t \leftarrow 0$
for $l \in L$ **do**
  $td \leftarrow td + \sqrt{(l.x - l_{old}.x)^2 + (l.y - l_{old}.y)^2}$
  if $l \in C$ **then**
    if $c = l_0$ **then**
      $d \leftarrow d + r_\tau$
    else if $c = l_{final}$ **then**
      if $\sqrt{(l.x - x_{old})^2 + (l.y - y_{old})^2} \geq 2r_\tau$ **then**
        $d \leftarrow d + r_\tau$
      else
        $d \leftarrow d + \sqrt{(l.x - x_{old})^2 + (l.y - y_{old})^2} - r_\tau$
      end if
    else
      if $\sqrt{(l.x - x_{old})^2 + (l.y - y_{old})^2} \geq 2r_\tau$ **then**
        $d \leftarrow d + 2r_\tau$
      else
        $d \leftarrow d + \sqrt{(l.x - x_{old})^2 + (l.y - y_{old})^2}$
      end if
    end if
    $x_{old} \leftarrow l.x$
    $y_{old} \leftarrow l.y$
  end if
  $l_{old}.x \leftarrow l.x$
  $l_{old}.y \leftarrow l.x$
end for
$p \leftarrow \frac{d}{td}$

O $(|R| \log(|R|))$. However, devices are mobile and we do not know, a priori, the time at which a new device will join the network. Therefore we cannot perform the sorting before it actually joins, especially since time is a continuous variable and we cannot simply sort the locations of all registered devices for all possible time values. As such, we can not reuse preprocessed results from one device for the other in the general case. The approach can also be regarded as a range query problem common in the Geographical Information Systems (GIS) [17] domain or mobile navigation domain. There are two categories of range computations: based on Euclidean distance, or based on road network distance. We are interested in the former. However, to the best of our knowledge, existing solutions in this category assume static objects of interest (e.g., [18, 19]). More attention has been paid to moving objects of interest in mobile object databases as done in [20, 21], whose approaches are complementary to our work. For instance, the work in [20] optimizes queries over moving objects by focusing on optimized process-

ing of location updates sent periodically by the mobile objects. We require no location updates from devices after they register but we can integrate this approach for a more accurate selection of services to access later on. However, we have to further analyze the tradeoffs between the increased accuracy of knowing the locations of devices and the increasing communication costs the updates will incur. In [21], authors propose an approach to optimize past queries over mobile objects. They do so by dividing each trip into sub trips depending on the transportation mode, which is out of the scope of our work. Nonetheless, their evaluations are of special interest to us as authors use a tool to simulate human mobility for $200,000$ mobile objects, which we can use in our future evaluations.

3. Probabilistic Registration Approach

As presented above, the deterministic registration approach provides a relatively precise decision but its time complexity increases linearly with the number of registered devices. Given that we target a large number of devices (millions or more), the approach can be very costly. To enhance its performance, we propose a probabilistic optimization that operates faster and still satisfies coverage threshold requirements. The optimization is built on using probabilistic mobility models to estimate the movements of mobile devices in a large network, essentially speeding up the spatial search phase of the algorithm. The resulting knowledge can then be exploited by an incoming device to make a decision whether or not to register its services. The decision is based on an estimate of the probability that any registered device with similar capabilities will cross paths with it. For this purpose, we choose the Truncated Lévy Walk (TLW) mobility model [12], which has been shown in recent work to well represent the mobility of humans (pedestrian, vehicles, etc.). TLW assumes that entities have a constant speed depending on the mobility category they belong to (pedestrians, vehicles, bicycles, etc.) [11]. In this model, entities select a uniformly distributed direction and a Lévy distributed length of displacement, after which they choose a Lévy distributed pause period. For simplicity purposes, we only consider devices belonging to the same mobility category.

3.1. Probabilistic Registration Approach Based on Direct Coverage

Detailed path information knowledge is now substituted by an estimation of the path of registered devices using TLW. As such, each device can use its mobility knowledge locally. We require the registry to provide the new device with basic information on the density of already registered devices and their distribution in space. For registered nodes, it is possible to estimate their spatial distribution along with its parameters, given their location, using curve fitting techniques (such as those provided by Matlab's `ALLFITDIST` method). As mentioned earlier, we need to compute the probability that the path of a new device hosting a sensor of type τ is covered. For this purpose, the approach should compute the probability that at least one device will be at each of the locations of interest at the same time as the new device. In this section, we focus on devices in R_τ. Let l_i be the location of the new device at time t_i and $P(\geq$ 1 device is at$(l_i, t_i))$ be the probability we are looking for at each location.

$$P(\geq \text{ 1 device is at } (l_i, t_i)) = 1 - P(\text{no device at } (l_i, t_i)) \quad (1)$$

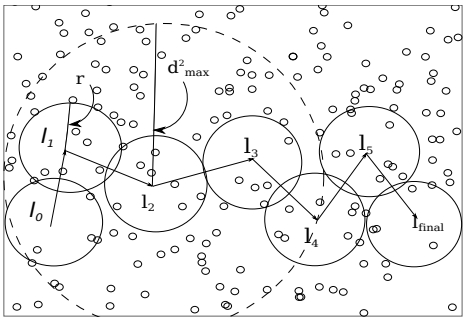

Figure 2: The big circle $C_{l_i,d^i_{max}}$ limits the range of devices that can reach l_2 in time t_2 while the smaller circles show the approximate location where sensors should be to consider that the location is covered

Note that we assume that the probabilities of different devices being at a location are independent. Consequently, $P(\text{no device at } (l_i, t_i))$ is as follows:

$$P(\text{no device at } (l_i, t_i)) = \prod_{\kappa \in R} (1 - P(\kappa \text{ is at } (l_i, t_i))) \tag{2}$$

The probability that device κ will be at location l_i at time t_i is the probability of device κ moving from its initial location to l_i. This refers to the total displacement δ_i^κ of device κ until time t_i from its initial location l_0^κ at $t = 0$. The probability to compute then becomes:

$$P(\geq \text{ 1 device is at } (l_i, t_i)) = 1 - \prod_{\kappa \in R} (1 - (P(\delta_i^\kappa) = l_i - l_0^\kappa)) \tag{3}$$

Using [11], where authors use TLW to compute the displacement of mobile entities for message routing purposes, we define the probability that device κ is in C_{l_i,r_τ} (to approximate l_i to an area covered by sensor of type τ instead of a point) at time t_i starting from a known location $l_0 = (X_0^\kappa, Y_0^\kappa)$ as

$$\begin{aligned} f(l_i, r_\tau, l_0, t_i) = & \quad P(\text{device } \kappa \text{ is in } C_{l_i,r_\tau} \text{ at time } t_i) \\ = & \quad \frac{1}{2\pi D t_i} \oint_{l^\kappa \in C_{l_i,r_\tau}} e^{\frac{(X^\kappa - X_0^\kappa)^2 + (Y^\kappa - Y_0^\kappa)^2}{2Dt_i}} dl^\kappa \end{aligned} \tag{4}$$

Where $l^\kappa = (X^\kappa, Y^\kappa)$, D is the diffusion factor in the Truncated Lévy Walk and it is equal to $\frac{\sigma_\xi^2}{\mu_t}$ [22], σ_ξ^2 is the variance of the displacement length and μ_t is the mean of the complete displacement time distribution. We consider σ_ξ^2 and μ_t to be parameters that depend on the real life scenario and the mobility category.

Since this approach does not use all locations of already registered devices, the initial location l_0^κ for device κ at time t_0 can be anywhere in the deployment area A. Going a step further, we can say that with respect to location l_i, we only care for devices located within the circle $C_{l_i,d^i_{max}}$ with center l_i and radius d^i_{max} (as shown in Figure 2); d^i_{max} is the distance from l_i beyond which no device κ can start from l^0_κ at t_0

and reach location l_i at time t_i. Taking into account all locations within circle $C_{l_i,d^i_{max}}$, the probability of device κ moving to C_{l_i,r_τ} becomes:[2]

$$g(l_i, r_\tau, t_i) = \oint_{l_0^\kappa \in C_{l_i,d^i_{max}}} PDF_{X_0^\kappa}(X^\kappa) * PDF_{Y_0^\kappa}(Y^\kappa) dl_0^\kappa \\ * \frac{1}{2\pi D t_i} \oint_{l^\kappa \in C_{l_i,r_\tau}} (e^{-\frac{(X^\kappa - X_0^\kappa)^2 + (Y^\kappa - Y_0^\kappa)^2}{2Dt_i}} dl^\kappa) \quad (5)$$

If we go back to the probability in Equation 3, we obtain by substitution:

$$P(\geq 1 \text{ device at } (l_i, t_i)) = 1 - \prod_{\kappa \in n} (1 - g(l_i, r_\tau, t_i)) \quad (6)$$

Where n is the set of all devices in $C_{l_i,d^i_{max}}$. The probability of displacement from l_0^κ into C_{l_i,r_τ} is identical for all devices in n, therefore $P(\geq 1$ device at $(l_i, t_i))$ becomes:

$$P(\geq 1 \text{ device at } (l_i, t_i)) = 1 - ((1 - g(l_i, r_\tau, t_i))^{|n|}) \quad (7)$$

Now that we have determined the probability of coverage at one location, we can repeat the process to obtain the complete probability of coverage at all locations.

$$P_{cov} = \prod_{l_i \in L} (1 - (1 - g(l_i, r_\tau, t_i))^{|n|}) \quad (8)$$

3.2. Probabilistic Registration Based On Coverage By Composition

If any location of interest on the device's path is not covered by sensors of its type τ, the device can check if its sensor can be substituted by a composition of other types of sensors (referred to as expansion types) at that location. It is important to assure that values are measured from substitution sensors at the same time instant and at the same location, i.e., by selecting sensors that are at an acceptable distance from each other. If we go back to our wind-chill example, we now need to compute the probability that all locations l_i on the path of the new device will be covered by both sensor types τ_1 and τ_2, i.e., sensor types $\in \ e^j_\tau$. In the following we show how we can compute the probability that takes composition into account.

Let $P^{S_\tau}_{cov}$ be the probability of coverage by sensor type τ, and $P^{e^j_\tau}_{cov}$ be the probability of coverage by all types in one of the expansion sets e^j_τ:

$$P^{e^j_\tau}_{cov} = \prod_{S \in e^j_\tau} P^S_{cov} \quad (9)$$

P^S_{cov} is computed using the same equation as P_{cov} (Eq. 8), with a change in the type of the sensor to evaluate. The complete probability of coverage, i.e., the probability including both direct coverage and coverage with composition cases, is:

[2]We multiply the density functions of X^0_κ and Y^0_κ by each other because we assume the location of the device on the X-axis is independent from its location on the Y-axis.

$$P_{covered} = \qquad P_{cov}(\text{ by } S_\tau \textbf{ or } e_\tau^j) \tag{10}$$

$P_{covered}$ is also equal to $1-$ (the probability of having no coverage by sensors of type τ and the probability of having no coverage by sensors in e_τ^j), i.e.,:

$$P_{covered} = \qquad 1 - ((1 - P_{cov}^{S_\tau}) * (1 - P_{cov}^{e_\tau^j})) \tag{11}$$

In a more general form, assuming there is a set E_τ of possible expansion sets, e_τ^j, the complete probability of coverage is as below:

$$P_{covered} = \qquad 1 - ((1 - P_{cov}^{S_\tau}) * \prod_{e_\tau^j \in E_\tau} (1 - P_{cov}^{e_\tau^j})) \tag{12}$$

Algorithm 4 Probabilistic Registration decision process for service of type t

Input: $(l_0, l_1, ..., l_L), r_\tau, E_\tau, R$
Output: $decision \in \{true, false\}$
1: $compute\ P_{covered}$ using Eq. 8
2: **if** $(P_{covered}) \leq threshold$ **then**
3: **for** $each\ e_\tau^j\ \in\ E_\tau$ **do**
4: **for** $each\ type\ \in\ e_\tau^j$ **do**
5: $compute\ P_{cov}$ using Eq. 8
6: **end for**
7: compute $P_{cov}^{e_\tau^j}$ using Eq. 9
8: **end for**
9: $p \leftarrow compute\ P_{covered}$ using Eq. 12
10: **end if**
// NT is the new threshold generated after computing the coverage probability
11: $NT \leftarrow 1 - \frac{(1-threshold)}{(1-p)}$
12: **if** $uniformRand(0, 1) < NT$ **then**
13: $decision \leftarrow true$
14: **else**
15: $decision \leftarrow false$
16: **end if**

We show in Algorithm 4 the method to generate the final registration decision based on the computed $P_{covered}$.

Complexity Analysis: The Probabilistic Registration algorithm terminates either if direct coverage is enough or when the end of the outermost loop is reached. The loop depends on the size $|E_\tau|$. There is one inner loop that depends on the size $|e_\tau^j|$, which is the number of expansion types in each set, also a finite number. Unlike the deterministic registration algorithm, which has to check the actual paths of registered nodes, the current algorithm is not dependent on those values. It depends on the size of the expansion set and expansion types which are constant and thus can be ignored. Computing $P_{covered}$ depends on $|L|$, which is a constant, and the areas $A_{C_{l_i,d_{max}^i}}$ for C_{l_i,d_{max}^i} and $A_{C_{l_i,r_\tau}}$ for C_{l_i,r_τ}, which can also be ignored. As such, our algorithm has a time complexity $\mathrm{O}\left(|E_\tau||e_\tau^j||L|A_{C_{l_i,d_{max}^i}} A_{C_{l_i,r_\tau}}\right)$ which reduces to $\Theta(1)$ since all values can be ignored. Note that steps 5, 7, and 9 incur additional time and space costs but those depend on the parameters of the numerical methods employed and are independent of the problem size. Further, the algorithm stores values in three sets. The first set is an array a passed when computing Eq. 8, containing the size of R_τ and the

type of their distribution in space. The second and third sets are E_τ and e_τ^j used when computing Eq. 12. Thus, the algorithm has a space complexity $\mathrm{O}\left(|E_\tau||e_\tau^j||a|\right)$ which reduces to $\Theta(1)$ since all three sets have fixed sizes and can be ignored.

4. MobIoT: Architecture and Implementation

To support different categories of participatory sensing applications and provide a scalable solution, we have designed and implemented MobIoT, a service-oriented middleware that incorporates the probabilistic registration approach discussed above. MobIoT further integrates semantic knowledge about the real world, including possible attributes to measure, their sensing units, mathematical relations among different attributes, etc. Consequently, developing a participatory sensing application no longer requires developers to be experts in the sensing domain and a user request can simply be specified in a query of the form:

```
new Query(new Selector(new Concept("temperature"), new
Concept("windspeed")), new Constraint(new Where("temperat-
ure.unit= Celsius, windspeed.unit= m/s, temperature.accur-
acy = 0.8")), new Location("Jardin des Tuileries"));
```

where `Concept` refers to the physical attribute to measure, `Selector` is used to specify the concept(s) of interest, `Constraint` is used to set the concept's unit and additional constraints such as the accuracy of a measurement, and finally `Location` allows the consumer to specify the location at which measurements should take place.

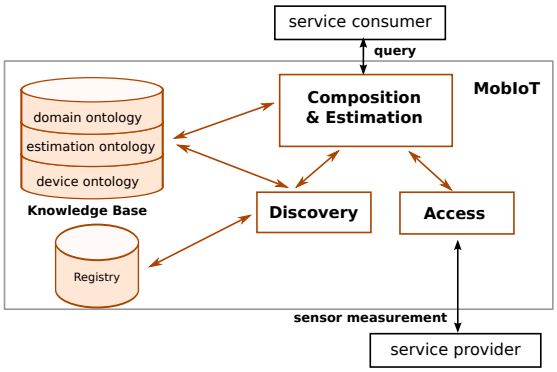

Figure 3: Architecture of MobIoT

4.1. MobIoT Architecture

The middleware is composed of the following five main components (Figure 3):

Knowledge Base. All knowledge required by our middleware for internal computations or by developers of sensing applications is defined in a set of ontologies, detailed in [16]. We build our ontologies on top of a set of NASA's SWEET ontologies (`http://sweet.jpl.nasa.gov/ontology/`) to model real world information, mathematics and physics, and so on. Our ontologies comprise:

- The *domain ontology*, describing physical concepts, measurement units, different possible relations among concepts (composition specification), etc. Composition specifications are defined as mathematical formulas that take measurements of several physical concepts as inputs, and return an estimation of the value of the physical concept of interest. For instance, the "wind chill" concept can be computed as a function of "temperature" and "wind speed".
- The *estimation ontology*, holding information needed for estimation techniques (such as linear interpolation, Kalman filter, naïve Bayesian learning), the equations that drive them, and the services that implement them.
- The *device ontology*, containing metadata about actual hardware devices, such as manufacturers, models, etc. This ontology allows us to determine what types of devices are needed to answer a user query.

Discovery. *Discovery* wraps service *Registration* and *Look-up* functionalities:

- The Registration functionality, detailed in Section 3, generates the final decision to allow or prevent a new mobile device from registering its hosted sensing services based on an estimate of whether or not registered mobile devices can cover a percentage of the new device's path. If the decision is positive, the service metadata is sent to the Registry for the latter to store it.
- The Look-up functionality returns a set of services that can provide the requested measurements at the location of interest.

Registry. In addition to holding the metadata and descriptions of registered sensing services, the registry executes the look-up, as it holds the information on the locations of devices when they register. The registry can thus estimate their location at the time of the query and know which of them best provides the required measurement.

Composition & Estimation. Expansions are specified as mathematical formulas in the domain ontology. The component then identifies the types of sensors that can measure the expansion concepts and discovers the addresses of devices hosting the identified types using the look-up functionality. Services are then accessed and their individual measurements are returned. All measurements over the same concept are aggregated based on an appropriate *fusion function* defined in the domain ontology. Finally, Aggregated results are passed as parameters to the expansion formula and the result is returned to the user.

Access. The access component allows local access to sensors hosted on mobile devices. It provides a unified interface for accessing any type of embedded sensor through one of the following options:

- *Instantaneous sensing* is a synchronous request that returns an immediate reply after querying a sensor for its latest sensing value.
- *Periodic sensing* is an asynchronous request that returns a reply at a constant rate (until canceled).
- *Event-based sensing* is an asynchronous request that returns a reply only when triggered by some event of interest (until canceled).

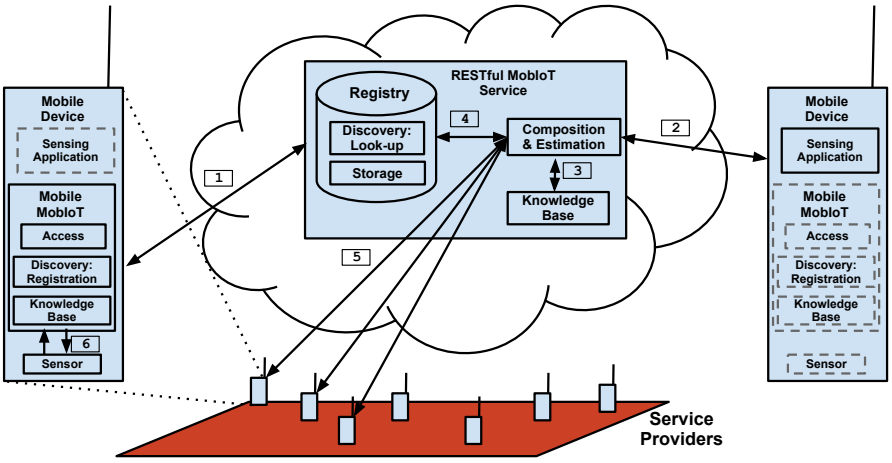

Figure 4: Middleware implementation.

4.2. Implementation

We implemented MobIoT using Java 1.6 in two parts as illustrated in Figure 4: Mobile MobIoT, a part of the middleware deployed on mobile devices (Android phones and personal computers), and a RESTful MobIoT service, a part of the middleware deployed as a RESTful J2EE Web service on a Tomcat servlet container. Both parts host a portion of the knowledge base locally. We model the ontologies using RDF (`http://www.w3.org/RDF`) and we access them using Jena (`http://jena.apache.org`) The code is part of CHOReOS, a European project on choreographies for the Future Internet (`http://www.choreos.eu`) [23] and is also used by several partners in their use case developments [24, 25]. Our code has been released as open source and links to the different components can be found at `http://choreos.eu/bin/Documentation/IoTS_Middleware` (in progress). We highlight in the following the main classes we created and the tools we used in our implementation. We present the methods, attributes and relations among the classes in UML in Figure 5.

4.2.1. Mobile MobIoT

Mobile MobIoT provides two main functionalities: *probabilistic registration* of sensors as services, and *access* to sensors (referring to the actual access to embedded sensors to acquire their measurements).

Firstly, the *probabilistic registration* functionality is provided through the cooperation of four Java classes:

- The `ExpansionSetGenerator` class identifies all possible expansion concepts based on the information defined in the local knowledge base. Expansion concepts are extracted from the local knowledge base using SPARQL, a powerful and expressive RDF query language.
- The `ProbabilityEstimator` class computes $P_{covered}$ as represented in Algorithm 4.

- The `RegistrationEstimator` class generates the actual registration decision based on $P_{covered}$ after receiving a registration request from a sensor service (step 1 in Figure 4).

Secondly, the *access* functionality is provided through the cooperation of three Java classes (step 6):

- The `SensorMediator` class is in charge of starting/accessing a Sensor Daemon, performing local discovery of sensors and calling sensing actions, etc.
- The `SensorDaemon` class is in charge of communicating with the sensors through Sensor Drivers.
- The `SensorDriver` class is an OS-specific piece of software that defines a small set of low-level methods for extracting sensor metadata and sampling a sensor.

4.2.2. RESTful MobIoT Service

The RESTful MobIoT service provides three main functionalities: *storage* (where sensing services' metadata is stored upon registration), *look-up* (where addresses of services that can answer a query are selected), and *composition & estimation* (where user queries are sent from a sensing application, compositions are identified and a final answer is returned).

Firstly, for the *storage*, we chose the Apache DERBY JDBC database as a backing store. We created a `Storage` Java class to save services, remove services, find number of registered services, etc.

Secondly, for the *look-up*, we created a `LookupManager` Java class that queries the backing store for the addresses of services that provide the required measurements at the location of interest (step 4).

Last but not least, the *composition & estimation* functionality is provided through the cooperation of four Java classes:

- The `QueryManager` class receives user queries from a sensing application and returns the final answer to the application (step 2). The query language is a union of SQL queries and TinyDB queries. It allows users to express sensing queries at locations of interest in a common language that handles sensing specific expressions such as sampling rate and sampling period.
- The `CompositionManager` class generates expansion concepts and executes expansion formulas.
- The `SPARQLManager` class handles the actual querying of the knowledge based to extract the expansion formulas, etc., using SPARQL query language (step 3). When expansion formulas are extracted from the ontologies, they are extracted as strings; to convert them to actual mathematical formulas we use Expr4j (Expr4j: http://expr4j.sourceforge.net/), an expression calculation engine for Java.
- The `AccessGenerator` class handles the remote access to sensing services hosted on mobile devices (step 5). We consider sensing services to be implemented as REST services with a unique address each. It is after calling the access method in this class, that the local access to sensors abstracted by the sensing services (in mobile MobIoT) is executed.
- The `DataFusionGenerator` class computes fusion/aggregation functions to combine measurements provided by different services.

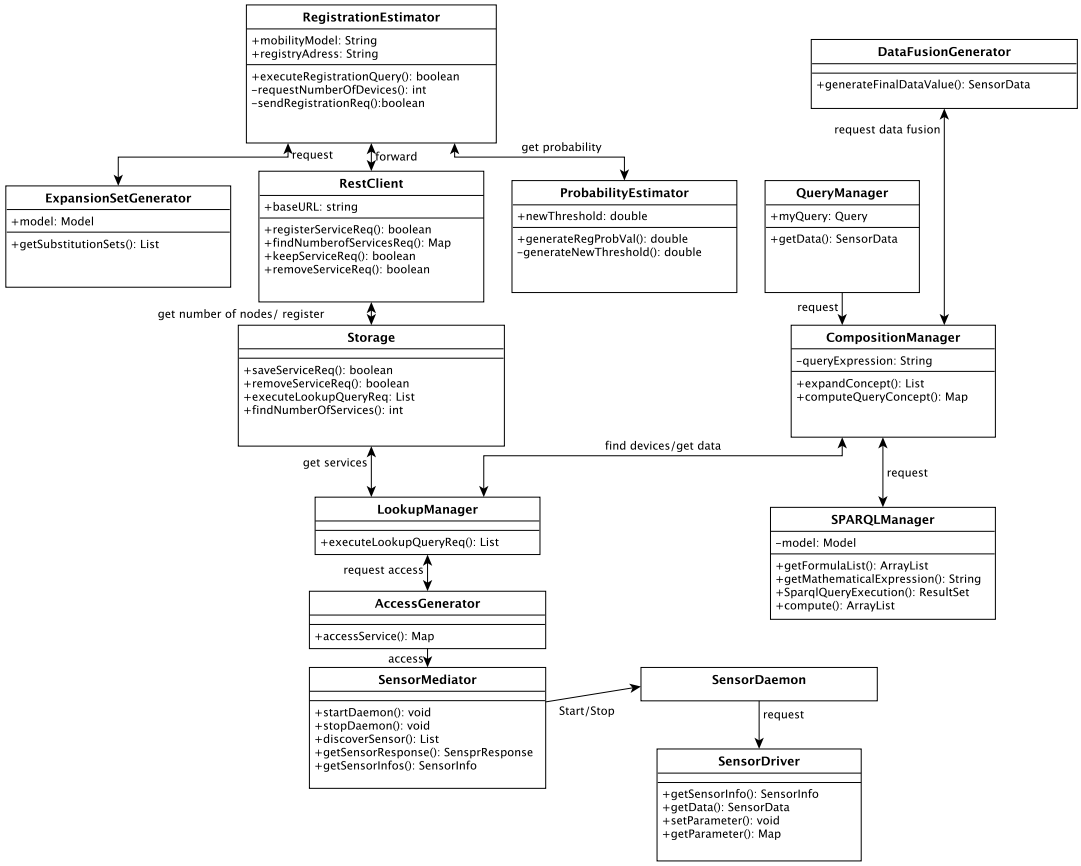

Figure 5: Class Diagram of MobIoT.

5. Evaluation

We present below the evaluation of our solution based on real mobility traces. We aim to show the validity of our probabilistic registration approach through the resulting coverage of the selected subset that matches the required coverage thresholds. We also evaluate the performance of both the deterministic and probabilistic approaches and compare the TLW-based computations to Random-Walk (RW)-based computations.

5.1. Experimental Setup

For a large scale realistic evaluation, we use a dataset that provides real mobility traces —in the form of (*taxi id, date time, longitude, latitude*)— for $10,000$ GPS-equipped taxis in Beijing [26]. We assume that a device that registers provides sensing services until the end of the evaluation. We set the length of the area A to 1136.2 km and breadth to 3002.4 km based on the area covered by the mobility traces. We computed the average velocity of those devices (8 km/hr) and set their sensing range to be

10 m.[3] We used this value of velocity to compute the value of the appropriate diffusion constant by simulating a Lévy Walk with a scale factor of 10 and 1 for the step-size and pause-duration respectively [11]. We chose the maximum allowed pause time to be $\tau_p = 1$ hr and the maximum allowed displacement length $\tau_\xi = 5$ km. The resulting diffusion value was found to be 44. The evaluation revolves around the following concepts:

- **Maximum Possible Coverage (MPC)**: Percentage of the area of interest A that is sensed/covered assuming all devices register their services as they show up.
- **Actual Coverage Percentage (ACP)**: Percentage of the MPC of A covered by the subset of devices that register their services.
- **Required Coverage Threshold**: The desired percentage of the area to sense (cover) by each device throughout its path.

5.2. Results

We evaluate the validity and performance of our registration approach in terms of three criteria: 1) How coverage varies as we shift from full registration (when all devices register their services) to probabilistic registration; 2) The time needed to generate a registration decision as the number of registered devices increases; 3) The appropriateness of the Truncated Lévy Walk model.

Coverage Quality: First, to evaluate how *coverage* varies as we shift from full to probabilistic registration with decreasing coverage requirements, we computed the MPC while the $10,000$ devices (the taxis in the trace) show up sequentially according to a poisson arrival process. The MPC when all $10,000$ devices register their services is 0.003% of the total area, which is very low, but we were unable to find larger datasets to evaluate our approach. Consequently, ACP values were also very low for the 0.8 and 0.6 thresholds. Given the sparsity of the devices, a large fraction will keep registering their services. The information is presented in Figure 6(a), which shows the ACP for 0.8 and 0.6 required thresholds, along with the registration curves for each of the thresholds as devices show up. The reason for the continuing registration is that the network has a low density and not all devices cross paths to substitute one another. We remind readers that ACP is a percentage of the MPC and not A, explaining why the MPC is presented as a 100% reference line in Figure 6(a). This is also the reason why, at the beginning of the evaluation, ACP curves start high and then decrease until they stabilize. More precisely, when the first device shows up, it will register, regardless of the threshold value, leading to: $ACP = MPC$ for a subset size of 1. Similar results will occur for the second device and so on until the thresholds start to effect the registration decisions as the sensing areas of devices start to overlap.

To better show the benefits of our approach, which are most relevant in highly dense networks, it is important to reach a MPC of 100% before all $10,000$ devices register. However, instead of adding phantom traces to increase the MPC of the set of devices, we decided to restrict the area of focus and increase the sensing range (we omit the details due to space constraints. More information can be found in [9]). The

[3]The sensing range was scaled to 10 m from the average sensing range of microphones in mobile phones (5 m, a value that we estimated empirically) carried by pedestrians who walk at an average speed of 4.5 km/h.

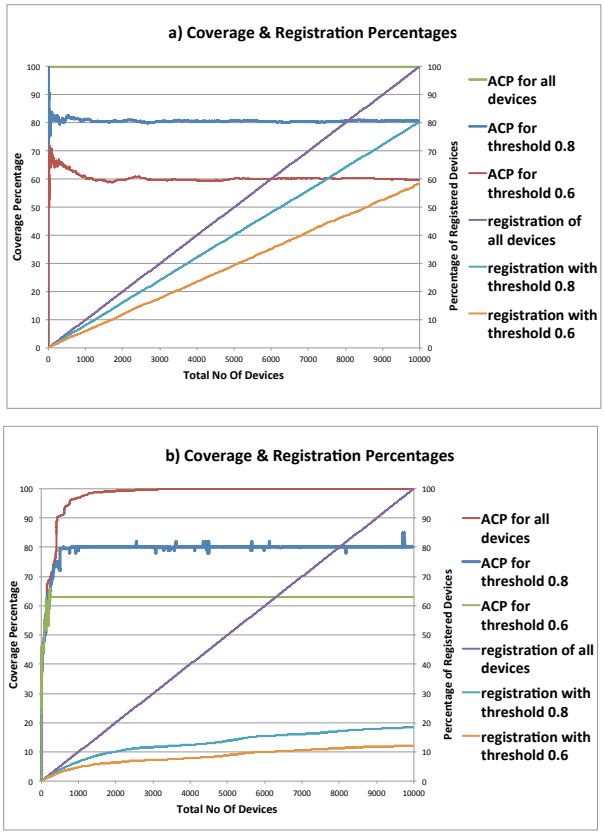

Figure 6: The resulting coverage and registration percentages as the required coverage threshold decreases from 1 to 0.6 a) for a radius of 10 m b) for a radius of 10 km.

chosen range is now 10 km. We are aware that the selected range is not practical for phone-based sensors but we used it to create a denser network that should lead to more overlapping sensed areas. The resulting MPC is shown in Figure 6(b). The figure illustrates how our approach successfully prevents devices from registering (the number of devices that register for a threshold of 0.8 is less than 2000) while still meeting the coverage requirements.

Computation Time: Second, to evaluate the *time needed for a service to register* as the number of registered devices increases, we deployed a process that iteratively computes registration decisions on a Linux machine with an Intel Xeon $X5650$ processor. We also deployed the Registry on a Linux Web Server with Apache Tomcat 6. We divide the registration time into two phases: the first, *local registration*, represents the time needed for the registration decision to be computed. The second phase, *web registration*, represents the time needed to actually register the service. We registered all the $10,000$ devices sequentially, which gives us the upper limit for registration time. The results (Figure 7) show us that registration times, for the probabilistic registration,

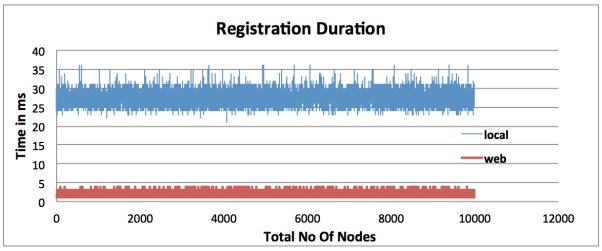

Figure 7: The time needed by the probabilistic registration approach for 10,000 devices to register with threshold = 1. Local represents the time needed to generate the registration decision and web represents the time needed to register services in the Registry.

remain between 2 and 4 ms and between 25 and 30 ms for first and second phases respectively. We also evaluated the time needed for the registration decision to be computed on mobile devices. We chose the Android Samsung Galaxy S3 with 1GB RAM. Results show that it takes, on average, 155 ms for the local process to be computed and 93 ms for the web process. It should be noted that the time taken in both local and web processes is independent of the subset size, which is a good sign of scalability.

We compared our probabilistic registration approach to the deterministic registration approach (with a similar setup as above) to verify, based on real mobility traces, that our probabilistic approach requires less computation efforts while satisfying the coverage threshold. Our claim is supported by the results, which show that it takes up to 2000 ms for a decision to register to be taken (Figure 8(a)). In most cases, it takes up to 400 ms (Figure 8(b)) for a decision to not register to be taken since the process stops as soon as the location on the new device's path are covered.

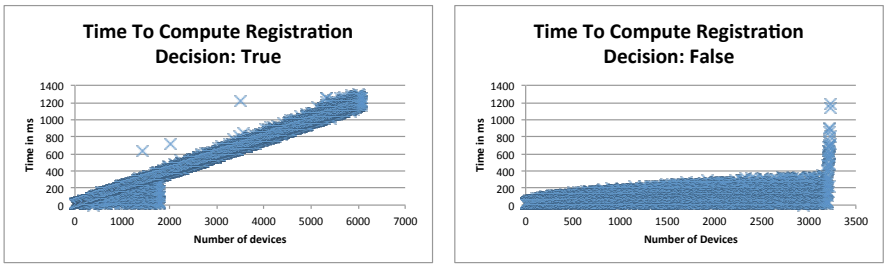

Figure 8: Time needed, by the deterministic registration approach with a threshold = 0.8, to decide to a) register b) not register.

TLW evaluation: Third, to evaluate the correctness of the Levy Walk Model, we compared the TLW-based results to Random-Walk (RW) based results, by substituting our probability computations with RW-based computations. Note that RW is also a commonly adopted model for estimating displacements of mobile entities. The results show that both models decrease the registration of devices while satisfying coverage requirements. However TLW outperforms RW as it leads to fewer devices participating (Figure 9). The figure shows the ACP of TLW- and RW-based registrations. As can be seen from the curves, the former leads to 760 devices registering while the latter leads

to 840 for a required threshold of 0.8 and 499 devices versus 639 for a threshold of 0.6. This means we have at least 10% fewer devices registering. Given the large scale we target, this is an important gain. It can also be seen that TLW-based registration reaches the required coverage threshold faster.

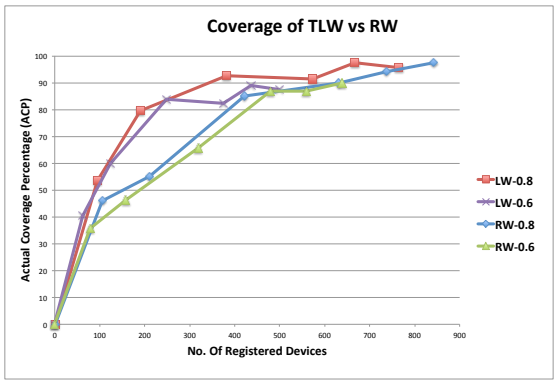

Figure 9: A comparison of the coverage/Registration percentage obtained by a TLW-based registration and a RW based registration for thresholds 0.8 and 0.6.

In conclusion, as the number of registered devices increases, our approach allows devices to register if need be (Figure 6(a)) and prevents them from doing so otherwise (Figure 6(b)) while not compromising the required coverage.

6. Related Work

The paper builds on the work presented in [9] where we first described the probabilistic registration solution. It extends the work by providing a deterministic solution that decreases the participation of mobile devices based on a more detailed knowledge of their future mobility. In addition to presenting both solutions, this paper compares their respective complexities and performances showing their benefits and drawbacks. We aim with our approach to limit the participation of mobile devices in participatory sensing tasks, in a manner that does not jeopardize the quality of the measurements. Existing works in the sensing domain have already applied this idea through duty cycling techniques, or device selection techniques where only some devices provide their sensing services while others sleep [14, 27]. However, duty cycling techniques are actually complementary to our work as they apply more to the look-up phase since devices are already known to the network.

Our work is similar in spirit to the solution in [28] where a recruitment framework is presented to select a subset of sensing devices based on their locations and coverage. However, there are major differences between the two approaches. In [28], the selection process depends on the willingness of users to participate, the coverage they provide and their reputation. The latter depends on the likelihood of users in providing the needed information when it is available and their timeliness in doing so, which can be complementary to our solution. Further, the approach requires a training period where participants' mobility information is collected, for a week, to know their

common routes, analyze their data and determine whether or not they are suitable to participate. This can be considered as an alternate approach to our required mobility knowledge. However, the selection of participants does not fit with our vision of a highly dynamic network as authors consider their participants to remain registered for longer periods of time. Finally, scale is not a main criteria as illustrated by the evaluation based on a panel of 30 users.

Further, most efforts in the participatory sensing domain have been restricted to the application level. More precisely, existing solutions are mostly domain-specific: environment monitoring (e.g., [29]), or traffic monitoring (e.g., [30, 31]), etc. Moreover, the sensing tasks, whether instantaneous, continuous or periodic, are usually not triggered by the query itself but programmed to wake up at some periods or after some event. Although there are several optimization techniques that can be applied such as historical data analytics to manage tradeoffs between data freshness and sampling frequencies, this approach may still produce large volumes of out-dated information. For instance, the work in [29] falls in the category of environment monitoring as the main objective is to model noise pollution in an area of interest as a noise map. The information is measured by smart mobile phones and then analyzed by a server to create a noise map that can be checked by users whenever needed. In [30], authors present a solution that falls in the category of traffic monitoring with an approach to estimate bus arrival times based on human input to localize the bus and allow the system to estimate its arrival time. Authors assume that there is at least one person in the bus or at a bus stop that updates the bus location information. Similarly, authors in [31] provide a solution to estimate bus arrival times; however they rely on a more passive approach where the bus is localized based on the nearest cell tower. All processing is done in a backend server. As a worst case scenario, the driver is equipped with a mobile phone that can provide data in case no other user wishes to contribute. The assumption used in both works — that it suffices to have one or two users provide their measurements— proves that in many cases, not all capable devices are needed to provide "good" information and in some cases even one device is enough.

We note that there are existing solutions that take the scale issue into account. Precisely, the scale issue is addressed from two perspectives: *User scale* when a large number of smart phone carrying users are involved in the participatory sensing process [32, 33]; and *data scale* when the sensing process generates large amounts of data that should be categorized, filtered out or condensed [34, 33]. Yet, in both cases, addressing the scale issue is always mostly performed at the data level, not the device level. For instance, authors in [34] present a scalable participatory sensing data delivery scheme through a location-based publish/subscribe framework. The framework is built on top of a location-based DHT overlay. The solution is geographically scalable but it was not evaluated with an increasing number of subscribers and publishers and therefore it is not clear how the scalability at the device level (i.e., regarding the number of devices) would be supported. In [35], scalability to allow the system to handle a large number (hundreds of thousands) of nodes is addressed by distributing the sensing tasks where, similarly to our work, participating nodes register their services to the server. The latter determines which tasks should be forwarded to which sensing device. However it is not clear how they deal with the resulting load on the server and they evaluate their work with simulated traces.

7. Conclusion and Future Work

We have presented in this paper a service-oriented middleware for large scale mobile participatory sensing. We focus in our work on query-based on-demand sensing that involves only a set of sensors that can best answer a query. To that end, we provide a novel registration approach that addresses the large scale and unknown network topology issues of mobile participatory sensing. The solution relies on the fact that mobile devices in dense networks are bound to cross paths and as such can substitute one another based on the (sensing) services they host. This approach requires knowledge of the mobility of registered devices and intersection computations are costly. To enhance the approach's performance, we replace the need to know the mobility of registered devices by a probabilistic mobility model to estimate the displacement of those devices. We evaluated the performance of both approaches. Results show that our probabilistic approach outperforms the deterministic approach as it successfully decreases the number of participating devices without decreasing the sensing coverage beyond an acceptable limit, while requiring less computation efforts.

Continuing our work, we plan on computing the coverage over the road network only instead of the whole area of Beijing thus ignoring the large spaces where taxis cannot drive, resulting in more accurate coverage results. We are also developing a probabilistic look-up approach that addresses the large number of mobile devices by selecting only a subset of registered devices based on the accuracy of the measurements they provide and the resources they have. The approach should also perform load balancing to prevent quick power depletion. Additionally, we are investigating alternatives for larger mobility traces for our evaluation work as we consider the size of the current trace ($10,000$ taxis) to be relatively small as compared to the millions of devices that can actually be part of participatory sensing tasks. Finally, we plan on integrating QoS criteria with our middleware for better service discovery, service composition and access.

Acknowledgment. This work is partially supported by the European Commission's Seventh Framework Programme FP7/2007-2013 under grant agreement number 257178 (project CHOReOS), the ANR-10-BLAN-0306 MURPHY project, and FP7-256980 NESSoS project. We would also like to thank Thiago Teixeira for his insight at early stages of this work.

References

[1] R. K. Ganti, F. Ye, H. Lei, Mobile crowdsensing: Current state and future challenges, IEEE Communications Magazine 49 (11) (2011) 32–39.

[2] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M. B. Srivastava, Participatory sensing, in: Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications, 2006, pp. 117–134.

[3] W. Khan, Y. Xiang, M. Aalsalem, Q. Arshad, Mobile phone sensing systems: A survey, IEEE Communications Surveys Tutorials 15 (1) (2013) 402–427.

[4] V. Issarny, N. Georgantas, S. Hachem, A. Zarras, P. Vassiliadist, M. Autili, M. Gerosa, A. Hamida, Service-oriented middleware for the future internet: state of the art and research directions, Journal of Internet Services and Applications 2 (1) (2011) 23–45.

[5] D. Raychaudhuri, M. Gerla, Emerging Wireless Technologies and the Future Mobile Internet, Cambridge University Press, 2011.

[6] M. Huhns, M. Singh, Service-oriented computing: Key concepts and principles, IEEE Internet Computing 9 (1) (2005) 75–81.

[7] M. Papazoglou, Service-oriented computing: concepts, characteristics and directions, in: Proceedings of the 4th International Conference on Web Information Systems Engineering., 2003, pp. 3 – 12.

[8] D. Maio, MSF: An Efficient Mobile Phone Sensing Framework, International Journal of Distributed Sensor Networks 2013.

[9] S. Hachem, A. Pathak, V. Issarny, Probabilistic registration for large-scale mobile participatory sensing, in: Proceedings of the 11th IEEE International Conference on Pervasive Computing and Communications, Percom'13, 2013.

[10] J. Krumm, E. Horvitz, Predestination: Inferring destinations from partial trajectories, in: Ubiquitous Computing (UbiComp'06), Vol. 4206 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 243–260.

[11] U. Sadiq, M. Kumar, Proximol: Proximity and mobility estimation for efficient forwarding in opportunistic networks, in: Proceedings of the 8th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, IEEE, 2011, pp. 312–321.

[12] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, S. Chong, On the levy-walk nature of human mobility, IEEE/ACM Transactions on Networking, TON'11 19 (3) (2011) 630–643.

[13] G. Y. Keung, Q. Zhang, B. Li, The delay-constrained information coverage problem in mobile sensor networks: single hop case, Wireless Networks 16 (7) (2010) 1961–1973.

[14] A. Ahmed, K. Yasumoto, Y. Yamauchi, M. Ito, Distance and time based node selection for probabilistic coverage in people-centric sensing, in: Proceedings of the 8th Annual IEEE International Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON'11, 2011, pp. 134–142.

[15] B. Wang, Sensor coverage model, Coverage Control in Sensor Networks (2010) 19–34.

[16] S. Hachem, T. Teixeira, V. Issarny, Ontologies for the Internet of Things, Proceedings of the PhD Student Symposium of the ACM/IFIP/USENIX 12th International Middleware Conference (2011).

[17] J. Star, J. Estes, Geographic information systems, prentice-Hall, 1990.

[18] M. A. Cheema, L. Brankovic, X. Lin, W. Zhang, W. Wang, Continuous monitoring of distance-based range queries, IEEE Transactions on Knowledge and Data Engineering 23 (8) (2011) 1182–1199.

[19] K. Xuan, G. Zhao, D. Taniar, B. Srinivasan, Continuous range search query processing in mobile navigation, in: Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems,ICPADS'08., IEEE, 2008, pp. 361–368.

[20] H. Mokhtar, J. Su, O. Ibarra, On moving object queries, in: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, 2002, pp. 188–198.

[21] J. Xu, R. H. Güting, Manage and query generic moving objects in secondo, Proc. VLDB Endow. 5 (12) (2012) 2002–2005.

[22] I. Rhee, M. Shin, S. Hong, K. Lee, S. Chong, On the levy-walk nature of human mobility, in: Proceedings of the 27th IEEE Conference on Computer Communications, INFOCOM'08, 2008, pp. 924 –932.

[23] CHOReOS consortium, CHOReOS Middleware 2nd Implementation (D3.2.2), Tech. rep. (2012). URL `http://www.choreos.eu/bin/Download/Deliverable`

[24] CHOReOS consortium, Passenger Friendly Airport Services Choreographies Design (D6.2), Tech. rep. (2011). URL `http://www.choreos.eu/bin/Download/Deliverable`

[25] CHOReOS consortium, DynaRoute Architectural Design (D8.2), Tech. rep. (2011). URL `http://www.choreos.eu/bin/Download/Deliverable`

[26] J. Yuan, Y. Zhen, X. Xie, G. Sun, Driving with knowledge from the physical world, in: In The 17th ACM SIGKDD international conference on Knowledge Discovery and Data mining, KDD'11, 2011.

[27] L. Chen, Y. Gu, S. Guo, T. He, Y. Shu, F. Zhang, J. Chen, Group-based discovery in low-duty-cycle mobile sensor networks, in: Proceedings of the 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON'12, 2012, pp. 542–550.

[28] S. Reddy, D. Estrin, M. Srivastava, Recruitment framework for participatory sensing data collections, in: Pervasive Computing, Springer, 2010, pp. 138–155.

[29] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, W. Hu, Ear-phone: an end-to-end participatory urban noise mapping system, in: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, ACM, 2010, pp. 105–116.

[30] B. G. Jagyasi, V. Kumar, A. Pande, Human participatory sensing in fixed route bus information system, in: Communication Technologies for Vehicles, Springer, 2012, pp. 113–123.

[31] P. Zhou, Y. Zheng, M. Li, How long to wait?: predicting bus arrival time with mobile phone based participatory sensing, in: Proceedings of the 10th international conference on Mobile systems, applications, and services, ACM, 2012, pp. 379–392.

[32] M. Von Kaenel, P. Sommer, R. Wattenhofer, Ikarus: large-scale participatory sensing at high altitudes, in: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, ACM, 2011, pp. 63–68.

[33] H. Lu, W. Pan, N. Lane, T. Choudhury, A. Campbell, Soundsense: scalable sound sensing for people-centric applications on mobile phones, in: Proceedings of the 7th international conference on Mobile systems, applications, and services, New York, NY, USA, 2009, pp. 165–178.

[34] H. Huang, E. Ngai, J. Liu, A location-based publish/subscribe framework for wireless sensors and mobile phones, in: IEEE Transactions on Wireless Communications and Networking Conference, WCNC'12, IEEE, 2012, pp. 2173–2178.

[35] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, A. Sharma, Prism: platform for remote sensing using smartphones, in: Proceedings of the 8th international conference on Mobile systems, applications, and services, ACM, 2010, pp. 63–76.