Privacy-Respecting Discovery of Data Providers in Crowd-Sensing Applications

Ioannis Krontiris
Goethe University Frankfurt,
Chair of Mobile Business & Multilateral Security,
Grueneburgplatz 1, 60323 Frankfurt, Germany
Email: ioannis.krontiris@m-char.net

Tassos Dimitriou
Computer Technology Institute, Patras, Greece
Computer Engineering Dept., Kuwait University, Kuwait
Email: tassos.dimitriou@ieee.org

***Abstract*—Crowd-sensing applications are based on the contribution of user-related context information and as such, they are particularly vulnerable to privacy-compromising attacks. In this paper we focus on the problem of information discovery by data consumers who can pose queries to mobile users providing sensed data. The way to protect the privacy of these mobile users is through the use of cloud-based agents, which obfuscate user location and enforce the sharing practices of their owners. The cloud agents organise themselves in a structure, namely a quadtree, that enables queriers to contact directly the mobile users in the area of interest and, based on their own criteria, select the ones to get sensing data from. The tree is kept in a decentralized manner, stored and maintained by the mobile agents themselves, thus avoiding the privacy implications of previous, centralized techniques. Our proposed solution complements and expands upon prior work in the area while it is shown experimentally to be both scalable, efficient and easy to maintain.**

***Keywords*-Crowdsensing; Mobile sensing; Privacy; Mobile cloud;**

I. Introduction

The increasing availability of sensors on today's smartphones and other everyday devices, carried around by millions of people, has already opened up new possibilities for gathering sensed information from our environment. Currently researchers experiment with these possibilities and share the vision of a sensor data-sharing infrastructure, where people and their mobile devices provide their collected data streams in accessible ways to third parties interested in integrating and remixing the data for a specific purpose. This trend is often named Participatory Sensing [1] and/or Mobile Crowdsensing [2]. A popular example is a noise mapping application which generates collective noise maps by aggregating measurements provided by the mobile phones of volunteers [3]. In other scenarios, people may monitor air pollution [4], road and traffic conditions [5], etc.

Several of the works on mobile sensing systems started differentiating very early between two data collection models. In the first model, users are actively involved in the collection process by deciding on the spot when to report data, while in the second model sensor sampling occurs whenever the state of the device (e.g. geographic location) matches the application's requirements described in a sensing task [6], [7]. But what is common in these architectures, is that the sensing data collected from the mobile phones are stored in a centralized server, where they are aggregated, processed and represented through various interfaces (e.g. statistical data on a map) or remain available for third parties to query and select data of interest [8].

Information collected from personal devices like mobile phones and tagged with GPS data is increasingly recognized as a kind of personal data, since it can reveal a lot about the person carrying the device [8], [9]. In addition to location, mobile sensing application make use of context information captured from the other sensors on the mobile device making the privacy implications even bigger. From this perspective, the collection of sensing data on service-side storage that the above architectures follow, creates fundamental asymmetries in the relationship between the users and the service providers and erodes transparency, confidence and trust.

To better protect user privacy, the user-centric approach is an alternative solution that gains recognition very rapidly over the last few years [10]. In our context, it would imply that sensing data is always handed back to the user upon completion of the service and remains under his control. It is the user's decision to hand over the data again to the same or another service. This way, individuals can supervise and limit personal data disclosure and exercise rights of access to their data held by third parties.

This approach has started to gain momentum recently in mobile sensing systems, with existing solutions suggesting a vault-like entity to provide an online trusted storage and processing [11], [12], [13], [14]. This vault, owned by an individual, allows users to login from their homes and define their own privacy policies and review/control who can see which kind of data, after their collection.

The problem that remains largely unexplored in this case is that of information discovery from data consumers. We refer specifically to the case where data consumers, i.e. queriers, either being applications or physical persons, are interested in retrieving information according to some requirements (location area, time frame, sensor type, etc.) from multiple data contributors that satisfy these requirements. That means they need to search data from data contributor's individual data stores, since there is not a central place where

all data is gathered. Given the distributed nature of data stores controlled by the corresponding data producers, this is not trivial to do. In addition, the querier might want to access specific data contributors based on criteria like their reputation gathered in previous participations, in order to guarantee some quality in the sensed data.

What makes the problem more challenging is the growing requirement of protecting queriers' data access privacy; a user may want to keep confidential whether (and when) she accessed the sensed data, the data types she was interested in, or from which nodes she obtained the data, as the disclosure of such information may be used to infer additional context about the user and used potentially against her interest.

So, in this paper, we suggest a mobile node discovering mechanism that corresponds to range queries about a geographical area. This mechanism enables, on one side, mobile users to protect their location privacy according to their own preferences and enforce access policies to their own data. On the other hand it enables queriers to contact directly the mobile users in the area of interest and select the ones to get sensing data from, based on their own criteria.

Our contributions are the following: *(i)* We use mobile agents on the cloud that take over the role of *representing* the mobile users to the outer world according to the user's location privacy preferences. *(ii)* The agents interconnect with each other in such a structure that enables queriers to discover mobile users in a specific geographic area. This structure is not stored by a central entity, but it is rather maintained by the cloud agents in a *distributed* fashion, thus avoiding the bottlenecks and the privacy implications of centralised approaches. *(iii)* Finally, we conduct extensive experiments demonstrating the efficiency of this approach in terms of scalability, load balancing and performance.

The rest of the paper is organised as follows. We first give an outline of related work in Section II. We then describe our system model in Section III, emphasising on the structure and functionality of the cloud agents. Section IV gives a detailed description of the operations required to maintain a distributed organisation of the agents in a structure that can serve range queries. Section V presents experimental results on the performance of this structure and, finally, Section VI concludes the paper.

II. Related Work

The use of stationary proxies has been already suggested in some participatory sensing systems so far, where they are used as data vaults or brokers for the user [11]. For example, Mun et al. proposed Personal Data Vault (PDV) [12], which functions as individual data storage with fine-grained access control mechanism, privacy rule recommender, and trace audit. Choi et al. also presented SensorSafe [13], an architecture that consists of multiple remote data stores and a broker enforcing a fine-grained access control by supporting privacy rules with context/behavior conditions and control for levels of inferences. What is common in all of the above systems is that the mobile phones sense and upload their data to their corresponding proxies *proactively*. Then the proxy is responsible to help the user manage this data and make it available to third parties, functioning as an access control mechanism. In our work, mobile nodes perform sensing operations only *reactively*, when there is a specific query to which they can respond to.

Drosatos et. al. [14] presented recently a privacy-respecting solution following the same reactive model, where mobile agents on the cloud store the data encrypted and execute a cryptographic protocol based on a homomorphic encryption scheme in order to aggregate the data and make them available. However, this setting does not consider the privacy of the querier, neither does it enable queriers to apply selection criteria on specific mobile nodes.

In our solution, cloud proxies interconnect with each other in a distributed structure, i.e. a quadtree, in order to enable queriers to discover data producers, while preserving the location privacy of those producers.

The use of trees has been used in a number of papers, however in a way that is mostly complementary to ours. For example, the authors in [15] use balanced trees to enforce k-anonymity in the spatial domain and conceal user locations (a user is considered k-anonymous, if its location is indistinguishable from that of $k-1$ other users). The emphasis here (and, similarly, in the work of [16] which make use of quadtrees) is on users *issuing* location based queries as in the case, for example, where a user asks for all hospitals close to its current location. As these queries may compromise user privacy, the use of trees is necessitated by the need to partition the geometric space and answer quickly queries about the location of other users in order to guarantee k-anonymity.

In our case, however, the users want to hide their location against a querier who is asking for data *provided* by the users. Through an appropriate decomposition of the space that is maintained in a *distributed* quad tree, as opposed to previous approaches which assume a centralized anonymizer, the users can easily obfuscate their exact location according to their *own* privacy preferences, without relying on other users as in the k-anonymity approaches.

III. System Model

Users carry personal sensors, either embedded in their mobile phones or part of wearable devices, and collect contextual data from their immediate environment. We will use the term *mobile object (MO)* to refer to these entities in the system who are actively sharing their sensing data with others. Each MO is also associated with a software agent running on a cloud, which operates on behalf of the MO and represents its interests.

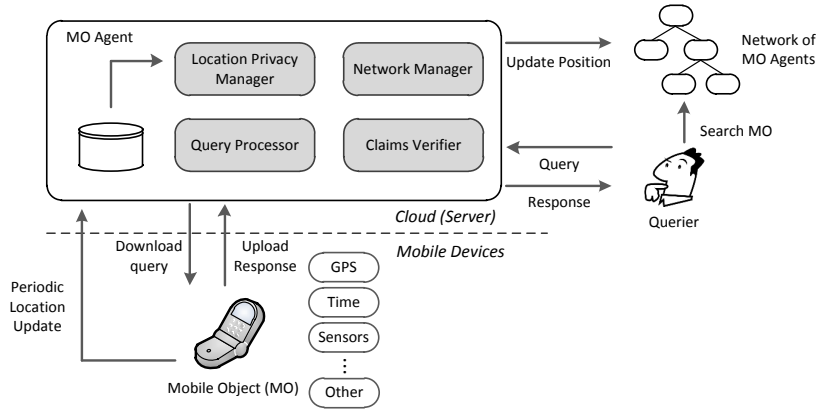


Figure 1: Query privacy in the context of participatory sensing.

A cloud agent serves as a proxy that extends and enhances the capability of the MO. As it resides on a stationary infrastructure, it has much higher availability and has not significant limitations on storage, communication and energy. It can thus offload tasks from mobile devices, that can be easily performed outside the device, and at the same time appear as the device's front-end to the rest of the world. This idea is well supported by the recent directions in cloud computing for resource-constrained mobile devices [17]. The device can upload personal information opportunistically to the agent, while the agent takes over the role of presenting these data to third-parties according to the owner's privacy preferences. In this way the device preserves its limited resources for the direct needs of its owner.

In our model, there is also the data requestor, or *querier*, who is interested in collecting data that satisfy some specific requirements. These requirements consist of the definition of a geographic area (range query), but also of additional parameters, like the kind of sensors needed, the time frame, etc. For simplicity, in this paper we assume that the geographic area is the only parameter, but the system can easily be extended to support additional ones. The goal of the system is to provide the querier with a list of all the MO agents that satisfy these requirements. Following that, the querier can select one or more agents, forward the query to them and receive the response.

The system preserves the privacy of the querier, by allowing her to stay anonymous (or pseudonymous) throughout the process. This means that the queries cannot be linked to the real identity of the querier. However, some access control mechanism is needed, so that not anybody can take benefit of the platform's services without demonstrating some sort of permission. How this permission can be obtained depends on the business model of the platform provider. For example, it could be that the querier has to pay for each "sensing quantum". To preserve the queriers privacy, the process of acquiring such a sensing quantum and the process of demonstrating it to the MO for enabling the processing of the query should be unlinkable with each other [18].

As we already mentioned in Section I, we want to enable the querier to target the list of MOs that move within the geographic region of interest, and therefore, we need to offer her some kind of "address book". In our solution, MO agents interconnect with each other and form a quadtree, preserving a hierarchy according to location and location precision they want to reveal. We will explain in the next section how this structure is being built and updated. What is important to emphasise here is that there is no specific central entity maintaining the topology information, but instead the network is distributed, with the agents storing and maintaining its structure.

Figure 1 depicts the general structure of a MO agent. The MO periodically updates its own agent about the changes of its location. Given that both entities are controlled by the user, this update can consist of accurate location information, depending on the technology used at the mobile device (GPS, Wi-Fi, etc.) The agent uses the received information to update its position in the quadtree. As we will see shortly, the agent does not reveal the location in full precision, but it obfuscates it according to the privacy preferences of the user. As a result, the MO agent resides in a place in the tree that corresponds to the given location and the given level of obfuscation.

Overall, an agent has several responsibilities with regard to its MO, as well as the maintenance of the network structure. The following sub-modules of the agent are responsible for these operations.

- Location Privacy Manager: It enables the user to set her location privacy preferences and obfuscates the location of the MO accordingly. The position of the agent in the network heavily depends on the precision of the location information that the MO is willing to reveal.

- Network Manager: Each MO agent takes some share from the task of network maintenance. They participate in building and hosting the information about the network topology and the links between the peers. The operations in this module are explained in Section IV.
- Claims Verifier: An access control mechanism is in place that protects the privacy of the querier on one hand, and on the other hand it enforces sharing policies defined by the user. These policies enable the user to control the extent to which her data is shared. In the general case, the querier presents claims to the MO agent satisfying the sharing policies of the user and the Claims Verifier is responsible for verifying these claims. If the process is successful, it forwards the query to the Query Processor. Privacy-ABCs [19] is a technology that can be used here to satisfy these requirements, in case different attributes of the querier need to be proved, but instantiations like in [18] can also be used in simpler cases.
- Query Processor: The MO agent is in charge of processing the received queries and enforce the location sharing policies set by the user. Such policies could restrict sharing for example to a specific time of the day, or within specific areas, etc. [20]. The filtering could also be based on the incentives offered by the querier.

IV. Distributed Quadtree-based Obfuscation

As mentioned above, the querier defines a geographic region and the goal of the system is to enable the querier to come in contact with the mobile nodes that are currently within this region. Additionally, the system allows users to obfuscate their location through their agents using spatial cloaking, as a way to protect their location privacy.

Obfuscation is the process of degrading the quality of information about a person's location, with the aim of protecting that person's location privacy. Most research to date has looked at the use of imprecision to degrade the quality of location information (e.g., [16], [21]). The concept of location obfuscation for mobile users was also studied within the EU-Project PICOS, where it was implemented and tested with actual users through extensive user trials. The experiments showed that the concept was actually the most desirable one amongst a set of measures that helped mobile users protect their privacy [22].

The need to serve spatial queries on one side and to enable location obfuscation on the other is best served by partitioning geographic space with a fixed space grid and use a region quadree structure as location data indexing method. In particular, the space is partitioned in two dimensions by decomposing the region into four equal quadrants, subquadrants, and so on until a predefined limit is reached. MOs obfuscate their location by spatial cloaking, that is by declaring their position based on these fixed quadrants, choosing the one with size closer to the obfuscation level they want. So, we move away from the typical solution of obfuscating location by defining a circle or square around the current location of the MO, which would move along with the MO and would make location profiling still possible. The choice of pre-defined quadrants as obfuscation areas better protect location privacy.

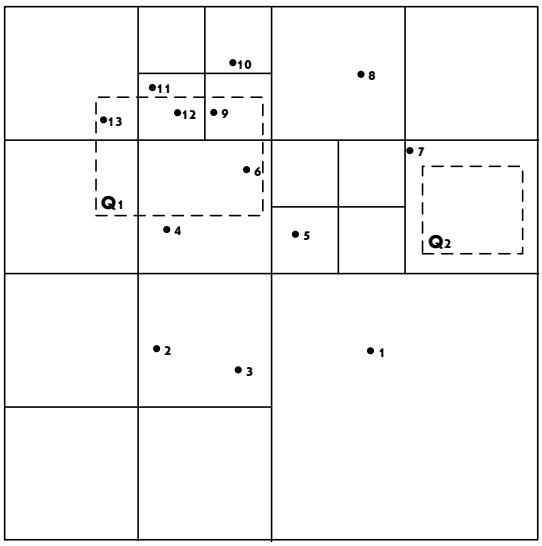
For example, Figure 2a shows an example of 13 mobile objects scattered in a region. Each of these MOs obfuscate its real location at different levels, choosing different size squares. So, for instance, MO_1 has the biggest level of obfuscation, while other objects like MO_{11} and MO_{12} declare their location being in more granular squares. This way of obfuscation implementation is different than classical solutions, where users are allowed to simply blur their location by declaring a region of a specific radius around their true location. Instead, here blurring is done using fixed squares of different sizes.

The above partition of space can be represented by a quadtree [23]. In the tree representation, the root node corresponds to the entire region we want to cover (e.g., a whole city). Each child of a node represents a quadrant (labelled in order NW, NE, SW, SE) of the region. As we go further down in the tree, nodes represent more accurate location information, until a predefined limit f_{max} (granularity) is reached. This limit f_{max} is predefined by the system and it corresponds to the maximum height of the tree. When a MO agent declares its position within a region, then it is also registered inside the corresponding tree node.

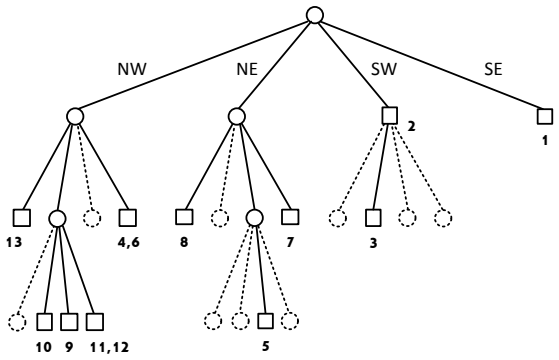
Each tree node stores two kinds of information: the addresses of the MOs registered by it and the pointers to its children. It can be the case that a node has no registered MOs and we call these kind of nodes *control nodes*. For example, Figure 2b shows the quadtree that corresponds to the region in Figure 2a. In the figure, a tree node with registered agents is represented with a square, while control nodes are represented with a circle. We have represented the missing children of a node with a dashed line. In this particular example of Figure 2, the maximal allowed location granularity is $f_{max} = 3$ and therefore the height of the corresponding quadtree is also 3.

There are two differences here, compared to the typical definition of a quadtree in the bibliography. First, we allow MOs to be stored in intermediate nodes and not just in the leaves. Second, we do not create a complete tree, i.e., if a leaf node has no registered MOs, we do not construct it.

What is important to realise is that the quadtree is not stored in a central place, but it is maintained in a distributed fashion among all participating MO agents (peers). Each peer stores its own share of the tree, while the tree decomposition is implicitly known by all the peers in the system, without the need for any communications. Below we explain in detail the basic operations on the quadtree, in order to make clear how it is structured and maintained.



(a) A region and its decomposition in quadrants. Mobile objects and queries are also depicted.



(b) Quadtree representation of the region in (a).

Figure 2: A region and its corresponding quadtree.

1) Initialize: At the beginning only the root node exists, which represents the whole city area. The root is the only node stored centrally in the service provider, so that everybody can use it as an entry point to the rest of the tree. We do not create any node in the tree, until a MO arrives that wants to register with the corresponding quadrant. In that case, the node is stored by that MO, according to the "insert" operation described below.

2) Insert: When a MO agent wants to join the network, it has to first identify the node that corresponds to its location and has the right level of granularity, according to its obfuscation level. So, starting from the the root it traverses the tree, till it reaches the node representing the desired region and registers with it.

In case the MO agent reaches a point where there is no node representing the next level in the path, it takes over the responsibility to store and maintain that node. It continues doing so, expanding the tree till it reaches the desired level. Then, it registers itself with the last node and remains responsible for maintaining the new part of the tree.

As an example, let us assume that MO with $Id = 5$, namely MO_5, is the first one that registers with the tree. Let us assume that MO_5 has defined its obfuscation level to be $f = 3$. That corresponds to level 3 in the quadtree. Figure 3 shows the exact position where MO_5 needs to register. Since the intermediate nodes under the root do not exist, MO_5 creates them together with the leaf node, where it registers itself and has responsibility for this part (shaded path). That means, the root node points to the address of MO_5 for the NE part of its subtree. Later, MO_7 and MO_8 join the same part, as we saw in Figure 2b. They only need

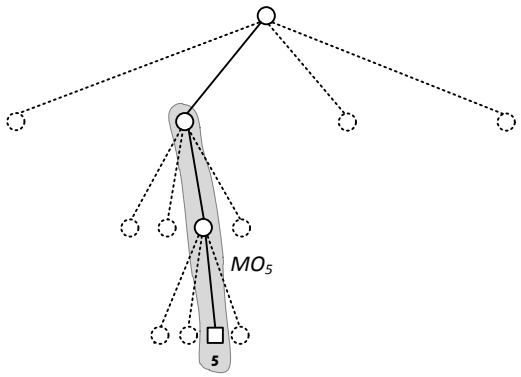


Figure 3: Insertion of MO_5 in an empty tree. The agent of the MO creates and stores the branch shown in the gray area.

to create and maintain the corresponding leaf nodes, where they register themselves.

3) Delete: MOs are always registered with the quadtree node that corresponds to their quadrant. When a MO wants to go offline, it needs to contact the MO that is responsible for maintaining that node and unregister. Also in the case the MO moves to another quadrant, a *Delete* has to be performed before the object registers with the new node, as we will see in the *Update* operation.

If a MO is responsible for maintaining some tree nodes, it keeps doing so, even if it goes offline and is not registered

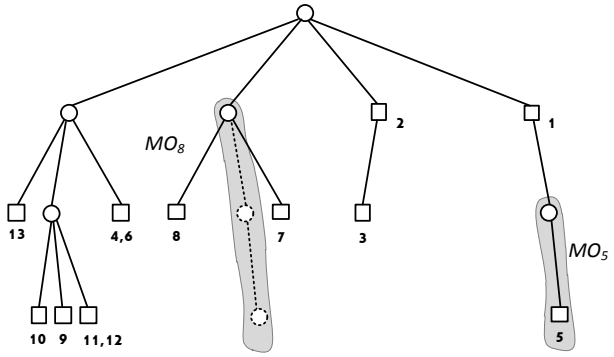


Figure 4: Example of an *Update*. MO_5 moves to the SE branch of the root. The grey area denotes the nodes stored by its agent.

with any node. The MO agent keeps running on the cloud and participates in maintaining the structure of the tree.

4) Update: As MOs move in the city, the quadtree is changing dynamically and MO agents need to update its structure. As long as a MO stays inside the same quadrant no updates are necessary, but once it moves outside it and into the next one, the following actions need to be performed. First, it needs to unregister from the previous quadrant, executing the *Delete* operation above. Then, it need to execute an *Insert* operation, as if the MO just joined the network. So overall, the update operation consists of the sequence of the previous two operations, i.e. a *Delete* and then an *Insert*.

Let us assume for example that MO_5 moves to the SE part of the map, close to MO_1, maintaining the same obfuscation level $f = 3$. As shown in Figure 4, MO_5 creates two new tree nodes and registers itself to the leaf node. It also keeps storing the three tree nodes from its previous position (see Figure 3), so in total it maintains 5 nodes, shown in the grey area of Figure 4.

The fact that an agent may be responsible for more than one part of the three has no impact on the MO itself. Since the agent is decoupled from the MO and acts as a proxy of it, it is not limited by the resource constraints of the MO.

5) Cleanup: As MOs move around and they change quadrants, it can be the case that some nodes in the quadtree are left with no registered MOs and with no children. In that case, the corresponding MO agent that stores this node can delete it from its memory. MO agents periodically invoke this check locally in their storage, as part of the quadtree maintenance. For example, in Figure 4, MO_5 stores two tree nodes (shown with a dashed line), which will be deleted as a result of two successive *Cleanup* operations.

6) Query: In order for a Querier to submit a query, she first needs to look up the nodes that might be able to answer the query. The procedure is very similar to the case where a new MO wants to join the tree. The Querier starts from the root and traverses the tree till she reaches the node that is representing the region of interest. The MO agent maintaining that node will provide the list of the registered MOs, possibly after verifying the corresponding credentials from the querier. Now she can communicate with the agents of MOs residing in the area of interest and see if they can help her with the query.

However, we allow a query region to also be an arbitrary region, as shown in Figure 2a by the dashed squares. In this case, the query region overlaps partially with several regions of the quadtree, which are at different levels. Since we don't know where exactly each MO is inside its region (as the user decides the level of obfuscation and the precision of its location), the query will have to return all MOs in the quadrants that overlap with the query. But this does not necessarily mean that they will *all* be inside the query region.

In the example shown in Figure 2a, the *Query* operation for Q_1 will return a list of all objects in the overlapping quadrants, including MO_4 and MO_{11}, even though these two are outside the query area. Any differentiation of those two MOs at this point would result in loss of their location privacy, since the querier, by asking several questions, could infer their exact location. Similarly, in the example of Q_2, there is actually no MO inside the query area, but the operation will return MO_7.

V. Experimental Results

To evaluate our algorithm of building and maintaining the quadtree, we implemented a simulator that takes as input the movements of MOs in the city over a period of time (corresponding to 100 location updates for each MO) and builds the corresponding quadtree. To generate a set of moving objects and their tracks, we used the Network-based Generator of Moving Objects [24]. As input to the generator we used the road network of two different cities, namely Oldenburg in Germany and San Joaquin in CA, USA. Oldenburg's road network, has a width of about $9\ Km$, while San Joaquin extends to $20\ Km$. So, choosing these two cities we cover the case of a small and a medium sized city for our simulations.

The output of the generator is a set of moving objects that move on the road network of the given city. The distribution of the moving objects in the spatial space is not uniform throughout the map, but it is correlated to the density of the network (i.e. the city centre is more busy than the outer region). In order to test the algorithms, we maintain the number of the moving objects in the city constant. The obfuscation level f of all moving objects is decided randomly following the uniform distribution and ranges between all levels of the quadtree (from the root until f_{max}). For simplicity, we assume that MOs do not change their obfuscation level.

1) Load-balancing and scalability: Figure 5 shows how the quadtree grows in number of nodes, as the population of MOs grows in the city. For Oldenburg we have assumed $f_{max} = 9$ and for San Joaquin $f_{max} = 10$. In this way, for both cities the quadtree extends to the same minimum supported quadrant size, equal to about $15\ m$. So, as it is expected, for San Joaquin, a bigger quadtree is needed compared to Oldenburg, for the same number of participating MOs.

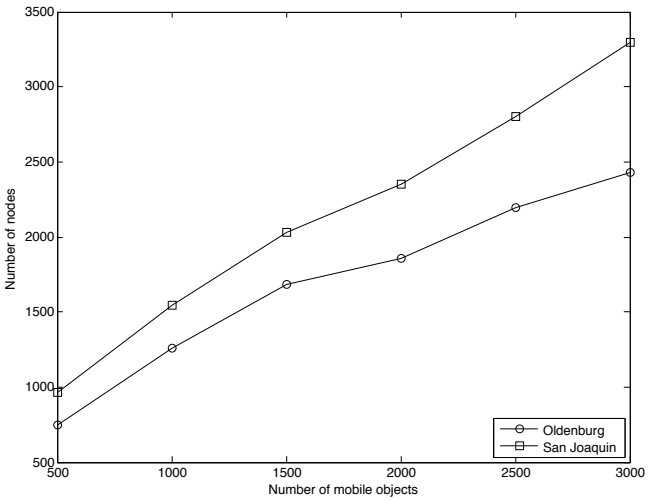


Figure 5: Size of the quadtree.

However, overall the number of nodes in the quadtree follows closely the number of MOs. This is shown in Figure 6 as well, where the average number of MOs registered with each tree node is depicted. In average it is not much more than one MO registered per node.

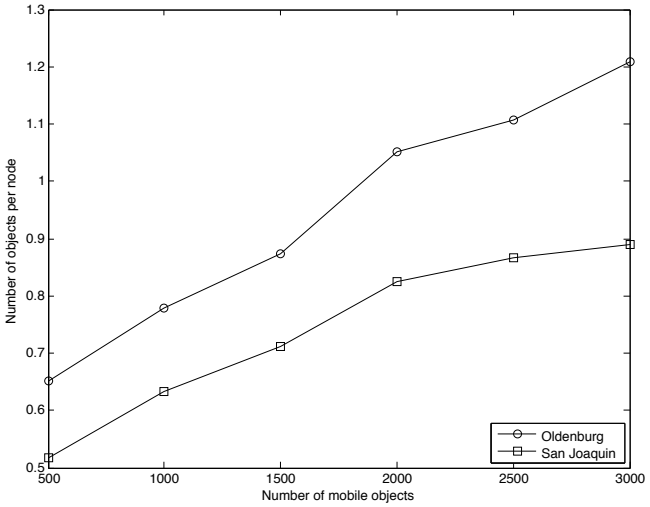


Figure 6: Number of MOs registered per tree node.

What is of particular importance here in terms of performance is the load placed on each MO for storing parts of the quadtree. Our simulations showed that the average number of nodes that each MO has to store is very low and remains unaffected by the total number of MOs. That is to be expected, since as the MOs increase, the size of the tree also increases (see Figure 5).

Figure 7 shows the probability mass function (pmf) of the number of quadtree nodes maintained by each MO agent for three different values of f_{max}. The experiment has assumed 500 MOs in Oldenburg. In all three cases one can see that the vast majority of agents store one or two nodes. For larger values of f_{max}, the tree is allowed to grow deeper and therefore it needs more nodes. So, the probability of two and three nodes per MOs is slightly increased.

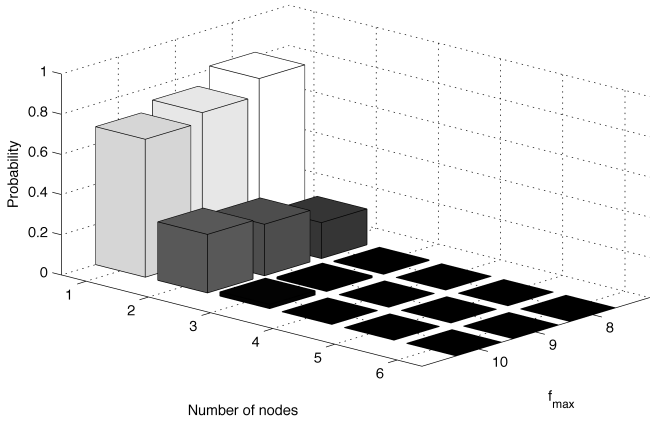


Figure 7: Probability mass function of the number of tree nodes maintained by each MO.

2) Updates: Figure 8 shows the frequency that a MO unregisters from the current node and registers in a new one, as it moves in the city. For both cities we have assumed 2000 participating MOs. The values in the y-axis are calculated as a percentage of the overall location updates sent by the MOs to their agents. So, for Oldenburg the MOs move to a new quadrant 37.3% of the time on average and their agent needs to update its position in the quadtree. As shown in the figure, this frequency can vary a lot and depends on how small the quadrants are allowed to become (i.e. on the value of f_{max}). This, however, does not have an impact on the MOs as the agents are separate entities living in the cloud and can easily handle the update operations.

It is also interesting to see the probability that during an update operation the tree node that corresponds to the new quadrant does not exist and the MO needs to create a new one. Let's take for example Oldenburg and assume 2000 MOs. Let's also assume that $f_{max} = 9$. Figure 9 shows the probability that at least one new node needs to be created in the quadtree as a result of an update operation. This probability depends at which level of the tree the MO moves. For nodes with $f < 4$ the probability is practically zero and it reaches up to 22% for nodes with $f = 9$.

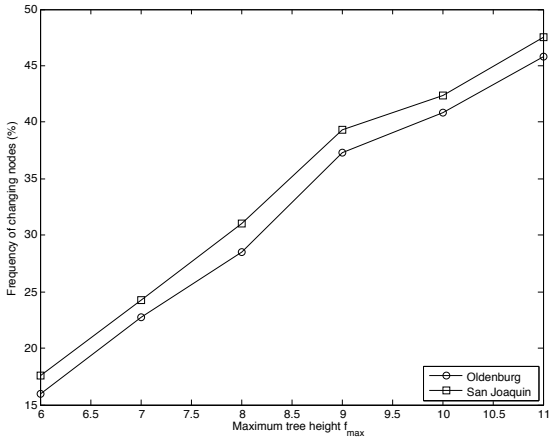


Figure 8: Frequency of MO update operations compared to the overall location updates.

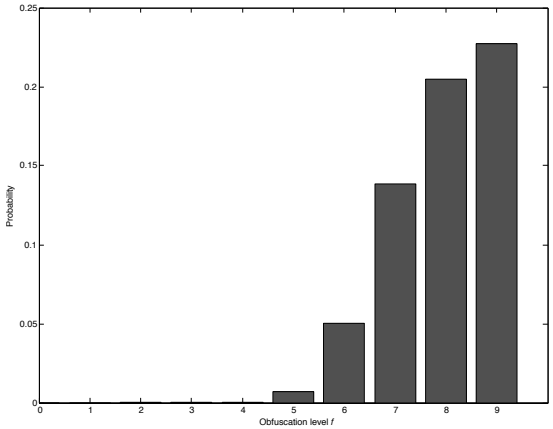


Figure 9: Probability of creating a new tree node as a result of a MO update operation.

3) Queries: We are interested to see what is the delay of performing a query. Since this is implementation dependent, we decided to use the simulations to measure a corresponding parameter that would give the scale of how the delay would vary, depending on the size of the query. Specifically we are interested to see what is the percentage of the whole tree that a querier has to traverse, in order to discover the MOs in the overlapping quadrants. For our experiment, we assume Oldenburg, with a population of 3000 MOs.

To perform the experiment, we post several queries, choosing some points randomly on the map and considering them as the centre of the queries' square. The size of the squares varies from large ones to smaller ones. In particular, we take nine different sizes that correspond to the size of the quadrants on nine different levels of the quadtree. In the case of Oldenburg taken in this experiment, level 1 in the tree corresponds to quadrants with diameter of $4.5\ Km$, level 2 to $2.25\ Km$, and so on, until level 9 where quadrants reach

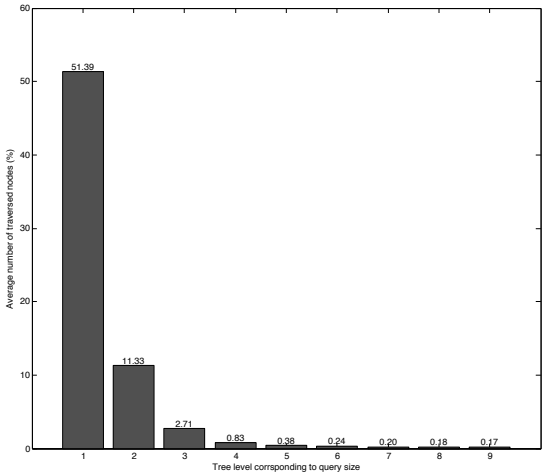


Figure 10: Percentage of traversed tree noes as a function of query size.

a diameter of $17.57\ m$. So, we experimented with queries of sizes that correspond to this scale (i.e. the tree levels).

Figure 10 shows what is the percentage of the total quadtree nodes that were traversed by the querier, depending on the size of the queries that she posts. By traversed we mean nodes that the querier visits as she executes the *Query* command described in Section IV. As the figure depicts, for the biggest queries, the percentage reaches up to 51.39%, but it drops by almost a factor of four for every next step. This is to be expected, since each level divides the space in four equal quadrants. So, a conclusion one can draw from Figure 10 is that the presented solution is efficient in case of queries that are targeted and more granular, compared to queries that cover large areas of the map.

VI. Conclusion

There is a tension between the need to serve spatial queries on one side and to enable mobile nodes protect their location privacy on the other. In this paper we have suggested partitioning geographic space with a fixed space grid and use a region quadree structure as location data indexing method in order to enable queriers come in contact with mobile nodes inside the area of interest. Mobile nodes are represented by software agents running on the cloud, which obfuscate location according to their users' privacy preferences and publish the result to the outer world.

Once the querier obtains a list of the mobile nodes inside the geographic area of interest, it can apply further selection criteria, e.g. based on reputation scores, to guarantee high quality responses. Similarly, the nodes can also apply selection criteria on the queriers, based on their own sharing preferences. This interaction, or rather "negotiation", between the two parties becomes even more challenging, if both are to maintain their privacy and we are currently working on it as part of our future work.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement no. 257782 for the project ABC4Trust.

REFERENCES

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

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile Crowdsensing: Current State and Future Challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[3] N. Maisonneuve, M. Stevens, and B. Ochab, "Participatory noise pollution monitoring using mobile phones," *Information Policy*, vol. 15, pp. 51–71, 2010.

[4] R. Honicky, E. A. Brewer, E. Paulos, and R. White, "N-smarts: networked suite of mobile atmospheric real-time sensors," in *Proceedings of the 2nd ACM SIGCOMM workshop on Networked systems for developing regions (NSDR '08)*, 2008, pp. 25–30.

[5] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys '08)*, 2008.

[6] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys '10)*, San Francisco, California, USA, 2010, pp. 63–76.

[7] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos, "AnonySense: A system for anonymous opportunistic sensing," *Journal of Pervasive and Mobile Computing*, 2010.

[8] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick, "A survey on privacy in mobile participatory sensing applications," *Journal of Systems and Software*, vol. 84, no. 11, pp. 1928 – 1946, 2011.

[9] I. Krontiris, F. C. Freiling, and T. Dimitriou, "Location privacy in urban sensing networks: Research challenges and directions," *IEEE Wireless Communications*, vol. 17, no. 5, pp. 30–35, October 2010.

[10] S. Fischer-Hübner, C. Hoofnagle, I. Krontiris, K. Rannenberg, and M. Waidner, "Online Privacy: Towards Informational Self-Determination on the Internet (Dagstuhl Perspectives Workshop 11061)," *Dagstuhl Manifestos*, vol. 1, no. 1, pp. 1–20, 2011.

[11] R. Cáceres, L. Cox, H. Lim, A. Shakimov, and A. Varshavsky, "Virtual individual servers as privacy-preserving proxies for mobile devices," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds (MobiHeld '09)*, Barcelona, Spain, 2009, pp. 37–42.

[12] M. Mun, S. Hao, N. Mishra, K. Shilton, J. Burke, D. Estrin, M. Hansen, and R. Govindan, "Personal data vaults: a locus of control for personal data streams," in *Proceedings of the 6th International COnference (Co-NEXT '10)*, Philadelphia, Pennsylvania, 2010.

[13] H. Choi, S. Chakraborty, Z. M. Charbiwala, and M. B. Srivastava, "Sensorsafe: a framework for privacy-preserving management of personal sensory information," in *Proceedings of the 8th VLDB International Conference on Secure Data Management (SDM '11)*, Seattle, WA, 2011, pp. 85–100.

[14] G. Drosatos, P. Efraimidis, I. Athanasiadis, E. D'Hondt, and M. Stevens, "A privacy-preserving cloud computing system for creating participatory noise maps," in *36th Annual IEEE Computer Software and Applications Conference (COMPSAC 2012)*, Izmir, Turkey, July 2012, pp. 581–586.

[15] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Prive: anonymous location-based queries in distributed mobile systems," in *Proceedings of the 16th international conference on World Wide Web (WWW '07)*, Banff, Alberta, Canada, 2007.

[16] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st international conference on Mobile systems, applications and services (MobiSys '03)*, San Francisco, California, 2003, pp. 31–42.

[17] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, January 2013.

[18] T. Dimitriou, I. Krontiris, and A. Sabouri, "PEPPeR: A Queriers Privacy Enhancing Protocol for PaRticipatory Sensing," in *Security and Privacy in Mobile Information and Communication Systems (MobiSec 2012)*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2012, vol. 107, pp. 93–106.

[19] ABC4Trust EU-Project. [Online]. Available: http://www.abc4trust.eu

[20] R. Schlegel, A. Kapadia, and A. J. Lee, "Eyeing your exposure: quantifying and controlling information sharing for improved privacy," in *Proceedings of the Seventh Symposium on Usable Privacy and Security (SOUPS '11)*, Pittsburgh, Pennsylvania, 2011.

[21] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Proceedings of the Third international conference on Pervasive Computing (PERVASIVE '05)*, Munich, Germany, 2005, pp. 152–170.

[22] M. Tschersich, C. Kahl, S. Heim, S. Crane, K. Böttcher, I. Krontiris, and K. Rannenberg, "Towards privacy-enhanced mobile communities - architecture, concepts and user trials," *Journal of Systems and Software*, vol. 84, no. 11, pp. 1947–1960, November 2011.

[23] A. Klinger and M. L. Rhodes, "Organization and access of image data by areas," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 1, pp. 50–60, 1979.

[24] T. Brinkhoff, "A framework for generating network-based moving objects," *Geoinformatica*, vol. 6, no. 2, pp. 153–180, Jun. 2002.