Available online at http://www.journalcra.com

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

International Journal of Current Research
Vol. 3, Issue, 4, pp.021-029, April, 2011

ISSN: 0975-833X

REVIEW ARTICLE

PRIVACY AWARE MONITORING FOR MOVING OBJECTS

*** Meyyappan, T. and Sangeetha, A.**

Department of Computer Science and Engineering, Alagappa University Karaikudi, Tamilnadu, India

ARTICLE INFO

Article History:

Received 8th February, 2011
Received in revised form
18th March, 2011
Accepted 5th April, 2011
Published online 17th April 2011

Key words:

Spatial databases,
Location-dependent and sensitive,
Mobile applications.

ABSTRACT

Efficiency and privacy are two fundamental issues in moving object monitoring. This paper proposes a privacy-aware monitoring (PAM) framework that addresses both issues. The framework distinguishes itself from the existing work by being the first to holistically address the issues of location updating in terms of monitoring accuracy, efficiency and privacy, in particular when and how mobile clients should send location updates to the server. Based on the notions of safe region and most probable result, PAM performs location updates only when they would likely alter the query results. Furthermore, by designing various client update strategies, the framework is flexible and able to optimize accuracy, privacy or efficiency. We develop efficient query evaluation/reevaluation and safe region computation algorithms in the framework. The experimental results show that PAM substantially outperforms traditional schemes in terms of monitoring accuracy, CPU cost and scalability while achieving close-to-optimal communication cost.

INTRODUCTION

In mobile and spatiotemporal databases, monitoring continuous spatial queries over moving objects is needed in numerous applications such as public transportation, logistics, and location-based services. Fig. 1 shows a typical monitoring system, which consists of a base station, a database server, application servers, and a large number of moving objects (i.e., mobile clients). The database server manages the location information of the objects.

****Corresponding author:*** meyslotus@yahoo.com; San.mphil@gmail.com

The application servers gather monitoring requests and register spatial queries at the database server, which then continuously updates the query results until the queries are deregistered. The fundamental problem in a monitoring system is when and how a mobile client should send location updates to the server because it determines three principal performance measures of monitoring-accuracy, efficiency, and privacy. Accuracy means how often the monitored results are correct, and it heavily depends on the frequency and accuracy of location updates. As for efficiency, two dominant costs are: the wireless communication cost for location updates and the query evaluation cost at the database server, both of which depend on the

frequency of location updates. As for privacy, the accuracy of location updates determines how much the client's privacy is exposed to the server. In the literature, very few studies on continuous query monitoring are focused on location updates. Two commonly used updating approaches are periodic update (every client reports its new location at a fixed interval) and deviation update (a client performs an update when its location or velocity changes significantly) (Jensen *et al.,* 2004; Pfoser and Jensen, 1999; Gedik and Liu, 2008; Ghinita *et al.,* 2007). However, these approaches have several deficiencies. First, the monitoring accuracy is low: query results are correct only at the time instances of periodic updates, but not in between them or at any time of deviation updates. Second, location updates are performed regardless of the existence of queries-a high update frequency may improve the monitoring accuracy, but is at the cost of unnecessary updates and query reevaluation. Third, the server workload using periodic update is not balanced over time: it reaches the peak when updates arrive (they must arrive simultaneously for correct results) and trigger query reevaluation, but is idle for the rest of the time. Last, the privacy issue is simply ignored by assuming that the clients are always willing to provide their exact positions to the server.

Some recent work attempted to remedy the privacy issue. Location cloaking was proposed to blur the exact client positions into bounding boxes (Gedik and Liu, 2008; Gedik and Liu, 2005; Hu *et al.,* 2005; Kalnis *et al.,* 2007). By assuming a centralized and trustworthy third-party server those stores all exact client positions, various location cloaking algorithms were proposed to build the bounding boxes while achieving the privacy measure such as k-anonymity. However, the use of bounding boxes makes the query results no longer unique. As such, query evaluation in such uncertain space is more complicated. A common approach is to assume that the probability distribution of the exact client location in the bounding box is known and well formed. Therefore, the results are defined as the set of all possible results together with their probabilities (Ghinita *et al.,* 2007; Chow *et al.,* 2006; Chen and Cheng, 2007). However, all these approaches focused on one-time cloaking or query evaluation; they cannot be applied to monitoring applications where continuous location update is required and efficiency is a critical concern.

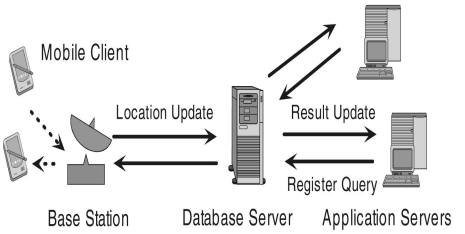

Fig. 1. The system architecture

In (Hu *et al.,* 2005), we proposed a monitoring framework where the clients are aware of the spatial queries being monitored, so they send location updates only when the results for some queries might change. Our basic idea is to maintain a rectangular area, called safe region, for each object. The safe region is computed based on the queries in such a way that the current results of all queries remain valid as long as all objects reside inside their respective safe regions. A client updates its location on the server only when the client moves out of its safe region. This significantly improves the monitoring efficiency and accuracy compared to the periodic or deviation update methods. However, this framework fails to address the privacy issue, that is, it only addresses "when" but not "how" the location updates are sent.

In this paper, we take a more comprehensive approach— instead of dealing with "when" and "how" separately like most existing work, we propose a privacy-aware monitoring (PAM) framework that incorporates the accuracy, efficiency, and privacy issues altogether. We adapt for the monitoring environment the privacy model that has been employed by location cloaking and other privacy-aware approaches. More specifically, a client encapsulates its exact position in a bounding box, and the timing and mechanism with which the box is updated to the server are decided by a client-side location updater as part of PAM. However, the integration of privacy into the monitoring framework poses challenges to the design of PAM. First, with the introduction of bounding boxes, the result of a query is no longer unique. Among all possible results, we argue that

the most probable result, i.e., the one with the highest probability, is most promising for approximating the genuine result (the result derived based on the exact positions). The probability is computed by assuming a uniform distribution of the exact client position in the bounding box. Fig. 2 shows two clients a; b together with their bounding boxes. Both the genuine and most probable result for the 1NN query Q are fag. However, even monitoring only the most probable result adds great complexity to query evaluation. As such, one of the main contributions of this paper is to devise efficient query processing algorithms for common spatial query types. Second, the most probable result also adds complexity to the definition of safe region. New algorithms must be designed to compute maximum safe regions in order to reduce the number of location updates, and thus, improve efficiency. Third, as the location updater decides when and how a bounding box is updated, its strategy determines the accuracy, privacy, and efficiency of the framework. The standard strategy is to update when the centroid of the bounding box moves out of the safe region, which guarantees accuracy—no miss of any change of the most probable result

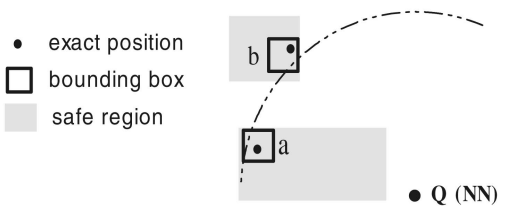

Fig. 2. Monitoring example

To optimize privacy or efficiency, however, alternative strategies must be devised. Compared to the previous work, the PAM framework has the following advantages:

- To our knowledge, this is the first comprehensive framework that addresses the issue of location updating holistically with monitoring accuracy, efficiency, and privacy altogether. This framework extends from our previous work (Hu *et al.,* 2005) by introducing a common privacy model, and therefore, suits realistic scenarios.
- As for efficiency, the framework significantly reduces location updates to only when an object is moving out of the safe region, and thus, is very likely to alter the query results.
- As for accuracy, the framework offers correct monitoring results at any time, as opposed to only at the time instances of updates in systems that are based on periodic or deviation location update.
- The framework is generic in the sense that it is not designed for a specific query type. Rather, it provides a common interface for monitoring various types of spatial queries such as range queries and kNN queries. Moreover, the framework does not presume any mobility pattern on moving objects.
- The framework is flexible in that by designing appropriate location update strategies, accuracy,privacy, or efficiency can be optimized.

In the rest of this paper, we will explore the PAM framework, especially on the aspects of query evaluation and safe region computation. The remainder of this paper is organized as follows: Section 2 reviews the related work. Section 3 overviews the framework components, followed by Sections 4 and 5 where query evaluation and safe region computation are presented, with an emphasis on range and kNN queries. Dynamic client update strategies are given in Section 6 to optimize privacy and efficiency. Experimental results of PAM are shown in Section 7.

II. RELATED WORK

There is a large body of research work on spatial temporal query processing. Early work assumed a static data set and focused on efficient access methods (e.g., R-tree) Guttman, 1984 and query evaluation algorithms (e.g., Hjaltason and Samet, 1999; Ghinita *et al.,* 2007). Recently, a lot of attention has been paid to moving-object databases, where data objects or queries or both of them move. Assuming that object movement trajectories

are known a priori. The Time-Parameterized R-tree (TPR-tree) for indexing moving objects, where the location of a moving object is represented by a linear function of time. Benetis *et al.* (2002) developed query evaluation algorithms for NN and reverse NN search based on the TPR-tree. Hu and. Xu, 2005, optimized the performance of the TPR-tree and extended it to the TPR_-tree. Chon *et al.* (2003) studied range and kNN queries based on a grid model. A novel index structure called STRIPES using a dual transformation technique. The work on monitoring continuous spatial queries can be classified into two categories. The first category assumes that the movement trajectories are known. Continuous Knn monitoring has been investigated for moving queries over stationary objects (Chon *et al.,* 2003) and linearly moving objects (Iwerks *et al.,* 2003). Iwerks *et al.* (2003) extended to monitor distance semi joins for two linearly moving data sets (Iwerks *et al.,* 2004). However, as pointed out in (Guttman, 1984) the known-trajectory assumption does not hold for many application scenarios (e.g., the velocity of a car changes frequently on road).

The second category does not make any assumption on object movement patterns. Xu *et al.* [24] and Zhang *et al.* [48] suggested returning to the client both the query result and its validity scope where the result remains the same. As such, the query is reevaluated only when the query exits the validity scope. However, their solutions work for stationary objects only. For continuous monitoring of moving objects, the prevailing approach is periodic reevaluation of queries (Jensen *et al.,* 2004; Khoshgozaran and Shahabi, 2007). Gedik and Liu (2008) proposed the Q-index, which indexes queries using an R-tree-like structure. At each evaluation step, only those objects that have moved since the previous evaluation step are evaluated on the Q-index. While this study is limited to range queries, Pfoser and Jensen (1999) proposed a scalable incremental hash-based algorithm (SINA) for range and kNN queries. SINA indexes both queries and objects, and achieves scalability by employing shared execution and incremental evaluation of continuous queries (Gedik and Liu, 2004; Iwerks *et al.,* 2003). Kalashnikov *et al.* 2004 suggested grid-based inmemory structures for object and query indexes to speed up reevaluation process of range queries and kNN queries. Access methods to support frequent location updates of moving objects have also been investigated (Jensen *et al.,* 2004). Our study falls into this category but distinguishes itself from existing studies with a comprehensive framework focusing on location update.

Uncertainty and privacy issues have been recently studied in moving object monitoring. To protect location privacy, various cloaking or anonymizing techniques have been proposed to hide the client's actual location. Among them are the spatiotemporal cloaking (Gruteser and Grunwald, 2003) the Clique-Cloak Gedik and Liu, 2005; Gedik and Liu, 2008 the Casper anonymizer hilbASR (Kalnis *et al.,* 2007), and peer-to-peer cloaking (Ghinita *et al.,* 2007a; Ghinita *et al.,* 2007b). In spatiotemporal cloaking, for each location update, the server divides the space Recursivel in a quad-tree-like format till a suitable subspace is found to cloak the updated location. The Clique Cloak algorithm constructs a clique graph to combine some clients who can share the same cloaked spatial area. The Casper anonymizer is associated with a query processor to ensure that the anonymized area returns the same query result as the actual location. In hilbASR, all user locations are sorted by Hilbert space-filling curve ordering, and then, every k users are grouped together in this order. Besides, location cloaking, pseudonym, dummy, and transformation were also proposed for privacy preservation. Pseudonym decouples the mapping between the user identity and the location so that an untrusted server only receives the location without the user identity (Beresford and Stajano, 2003). Dummy generates fake user locations (called dummies) and mixes them together with the genuine user location into the request (Kalnis *et al.,* 2007; Kalashnikov *et al.,* 2004). Transform ation utilizes certain one-way spatial transformations (e.g., a space filling curve) to map the query space to another space and Resolves query blindly in the transformed space (Khoshgozaran and Shahabi, 2007). As for location uncertainty, a common model for characterizing the uncertainty of an object is a closed region with a predefined probability distribution of this object in the region. Based on this probabilistic model, query processing and indexing algorithms have

been proposed to evaluate probabilistic range queries and kNN queries (Cheng *et al.,* 2004). While in these studies, the objects are uncertain, the queries themselves are still certain. Chen and Cheng extended the probabilistic processing to more general cases where the queries are also uncertain (Chen and Cheng, 2007). Our study, on the other hand, addresses the continuous monitoring issue. By adopting the notion of "safe region," the frequency of query reevaluation on uncertain location information is reduced, and hence, the system efficiency and scalability are improved. Distributed approaches have been investigated to monitor continuous range queries (Cai *et al.,* 2004) and continuous kNN queries (Pfoser and Jensen, 1999). The main idea is to shift some load from the server to the mobile clients. Monitoring queries have also been studied for distributed Internet databases (Chen *et al.,* 2000), data streams (Babu and Widom, 2001) and sensor databases (Hjaltason and Samet, 1999). However, these studies are not applicable to monitoring of moving objects, where a two-dimensional space is assumed.

III. Fundamentals of PAM framework

A) Privacy-Aware Location Model

In this paper, we assume that the clients are privacy conscious. That is, the clients do not want to expose their genuine point locations to the database server to avoid spatiotemporal correlation inference attack (Gedik and Liu, 2005) by which an adversary may infer users' private information such as political affiliations, alternative lifestyles, or medical problems. For example, knowing that a user is inside a heart specialty clinic during business hours, the adversary can infer that the user might have a heart problem. This has been cited as a major privacy threat in location-based services and mobile computing. To protect against it, most existing work suggests replacing accurate point locations by bounding boxes to reduce location resolutions (Chen and Cheng. 2007; Ghinita *et al.,* 2007). With a large enough location box covering the sensitive place (e.g., the clinic) as well as a good number of other insensitive places, the success rate or confidence of such spatiotemporal correlation inference can be reduced significantly. In our monitoring framework, we take the same privacy-aware approach. Specifically, each time a client detects his/her genuine point location; it is encapsulated into a bounding box. Then, the client-side location updater decides whether or not to update that box to the server.1 Without any other knowledge about the client locations or moving patterns, upon receiving such a box, the server can only presume that the genuine point location is distributed uniformly in this box. To simplify the presentation in thispaper, we further restrict the shape of such a bounding box to a _-by-_ square (or in short _-square), where _ is customizable for each object. Our problem is therefore to monitor result changes of spatial queries as objects move, and monitor them as accurately as possible and at the lowest cost of location updates. The key idea to solving the problem is "safe region," which was defined in (Hu *et al.,* 2005) as a rectangle within which the change of object location does not change the result of any registered spatial query. Now that locations are _-squares instead of points, to clarify the definition of "within," we use the centroid point of the square as a representative, so the safe region is essentially a safe region for the centroid of the _-square. However, the consequence of introducing _-square is more than that—the result of a spatial query is no longer unique. For example, if the _-square of an object partially overlaps with a range query, this object could be either a result object or a nonresult object of this query. As such, a unique definition of query result under _-squares is a prerequisite of safe region. Since the genuine point location of an object is distributed uniformly in its _-square, we can define the (unique) query result as the one with the highest probability among all possible results. As in the previous range query example, if the majority of the _-square falls inside the range query, that object is most probably a result object of this query; otherwise, that object is most probably a nonresult object. With the notion of most probable result, we thereby define the safe region as a rectangle within which the change of the centroid of the object's _-square does not change the most probable result of any registered spatial query. The standard update strategy of the client is therefore "to update when the centroid of the -square is out of the safe region." The reason why we exclude all other less probable results in this definition is threefold: 1) monitoring

continuous queries usually trades accuracy for efficiency — although the most probable result does not always align with the genuine result (the result derived based on genuine point locations of all objects), we will show in Section 4 that it is efficient to compute, and therefore, prevents the server from being computationally overloaded; 2) if the query result were defined as the set of all possible results, the safe region would have to be extremely small to report location updates if any of t he possible results changes, which makes the update cost overwhelmingly high; and 3) we do not want the choice of _-square—which is made by the client —to affect query results heavily, and obviously the most probable results are less vulnerable than other result definitions.

B) Framework Overview

As shown in Fig. 3, the PAM framework consists of components located at both the database server and the moving objects. At the database server side, we have the moving object index, the query index, the query processor, and the location manager. At moving objects' side, we have location updaters. Without loss of generality, we make the following assumptions for simplicity:

- The number of objects is some orders of magnitude larger than that of queries. As such, the query index can accommodate all registered queries in main memory, while the object index can only accommodate all moving objects in secondary memory. This assumption has been widely adopted in many existing proposals (Gedik and Liu, 2008; Khoshgozaran and Shahabi, 2007).

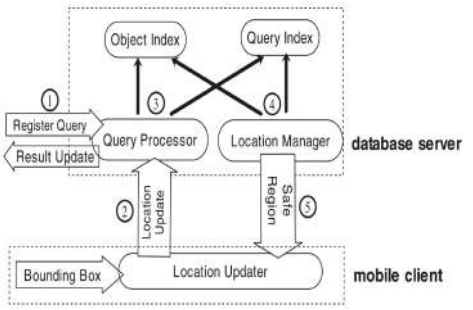

Fig. 3. PAM framework overview

- The database server handles location updates sequentially; in other words, updates are queued and handled on a first-come-first-serve basis. This is a reasonable assumption to relieve us from the issues of read/write consistency.
- The moving objects maintain good connection with the database server. Furthermore, the communication cost for any location update is a constant. With the latter assumption, minimizing the cost of location updates is equivalent to minimizing the total number of updates.

PAM framework works as follows (Fig. 3): At any time, application servers can register spatial queries to the database server (step _1). When an object sends a location update (step_2), the query processor identifies those queries that are affected by this update using the query index, and then, reevaluates them using the object index (step _3). The updated query results are then reported to the application servers who register these queries. Afterward, the location manager computes the new safe region for the updating object (step _4), also based on the indexes, and then, sends it back as a response to the object (step _5). The procedure for processing a new query is similar, except that in step _2 , the new query is evaluated from scratch instead of being reevaluated incrementally, and that the objects whose safe regions are changed due to this new query must be notified. Algorithm 1 summarizes the procedure at the database server to handle a query registration/deregistration or a location update.

It is noteworthy that although in this paper, the most probable result is used, this framework can also adapt to other query result definitions such as over a probability confidence (e.g., "returns objects that have 90 percent probability inside the query range"). The only changes needed to reflect the new result definition are the query evaluation algorithms in the query processor and safe region computation in the location manager. In the rest of this paper, we stick to the definition of the most probable result and leave the modification details for other definitions to interested readers. The

following sections explain the components at the database server in detail, and Section 6 describes the update strategy of the client-side location updater.

IV. Query processing

In this section, we present the detailed algorithms to evaluate or reevaluate a spatial query Q in terms of the most probable result. Aside from the definition of the query result, we know that Q also differs from a conventional spatial query in that the object locations are in the form of _-square (for updating objects) or bbox (for other objects), both of which are rectangular. In this section, instead of regarding Q as a special query type, we take an alternative approach by regarding the space where the object locations are defined as a special Euclidean space. In this space, spatial relations such as overlapping, containment, or even distance are implemented differently from a conventional Euclidean space. By using the new implementations of spatial relations, existing spatial query processing algorithms can be applied directly to the new space. In the following sections, we implement two relations that are required for spatial queries, namely, containment and closer.

Spatial Relations

In this new space, an object p is contained in a rectangle R if in the Euclidean space, the majority of p is in R. The rectangle divides the enlarged safe region of any object p into two regions: the region inside rectangle q (where p is a result object of q) and the region outside q (where p is not a result). The region with the larger area decides the most probable result. In this new space, an object p1 is closer to a point q than object p2 if and only if in the Euclidean space, for two randomly picked points a; b from p1 and p2, respectively, as is equally or more probably closer to q than b. The closer relation has a nice property that it is a total order relation.

Scalability

This section evaluates the scalability of all frameworks in terms of the server's CPU time and communication cost. The CPU time when the number of registered queries (W) increases from 10 to 1,000 (Fig. 4). PAM only increases by less than 10 times because the grid-based query index filters out most of the unaffected and irrelevant queries. However, for PRD (1) and PRD (0.1), the CPU time is linear to W, as they need to reevaluate every query at each batch of location updates. When W ¼ 1;000, for one logical time unit, the server needs 1.6 CPU seconds to monitor the 100,000 moving objects using PAM, 53 seconds using PRD (1), and 217 seconds using PRD (0.1). As PRD updates locations periodically, the high CPU cost imposes on it a

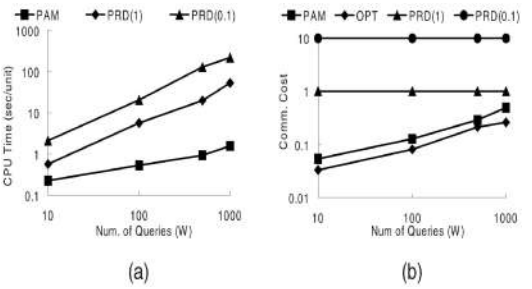

Fig.4. Performance versus query numbers a) CPU time. b) Communication cost.

maximum update frequency: in this example, the update frequency is at most once every 21.7 seconds. PAM has no such limitation. In terms of communication cost, although PAM increases linearly with respect to W, it is still less than double of OPT. All the above results suggest that PAM is robust under various W settings.

Conclusion

This paper proposes a framework for monitoring continuous spatial queries over moving objects. The framework is the first to holistically address the issue of location updating with regard to monitoring accuracy, efficiency, and privacy. The performance of our framework is evaluated through a series of experiments. The results show that it substantially outperforms periodic monitoring in terms of accuracy and CPU cost while achieving a close-to-optimal communication cost. Furthermore, the framework is robust and scales well with various parameter settings, such as privacy requirement, moving speed, and the

number of queries and moving objects. As for future work, we plan to incorporate other types of queries into the framework, such as spatial joins and aggregate queries. We also plan to further optimize the performance of the framework. In particular, the minimum cost update strategy shows that the safe region is a crude approximation of the ideal safe area, mainly because we separately optimize the safe region for each query, but not globally. A possible solution is to sequentially optimize the queries but maintain the safe region accumulated by the queries optimized so far. Then, the optimal safe region for each query should depend not only on the query, but also on the accumulated safe region.

REFERENCES

Babu, S. and Widom, J. 2001. Continuous Queries over Data Streams. Proc. ACM SIGMOD.

Beckmann, N., Kriegel, H., Schneider, R. and Seeger, B. 1990. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. Proc. ACM SIGMOD, pp. 322-331.

Benetis, R., Jensen, C.S., Karciauskas, G. and Saltenis, S. 2002. Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects," Proc. Int'l Database Eng. and Applications Symp. (IDEAS),

Beresford, A. and Stajano, F. 2003. Location Privacy in Pervasive Computing," IEEE Pervasive Computing, vol. 2, no. 1, pp. 46-55.

Broch, J., Maltz, D.A., Johnson, D., Hu, Y.C. and Jetcheva, J. 1998. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," Proc. ACM/IEEE MobiCom, pp. 85-97.

Cai, Y., Hua, K.A. and Cao, G. 2004. Processing Range-Monitoring Queries on Heterogeneous Mobile Objects," Proc. IEEE Int'l Conf. Mobile Data Management (MDM).

Chen, J. and Cheng, R. 2007. Efficient Evaluation of Imprecise Location-Dependent Queries," Proc. IEEE Int'l Conf. Data Eng. (ICDE), pp. 586-595.

Chen, J., DeWitt, D., Tian, F. and Wang, Y. 2000. NiagaraCQ: A Scalable Continuous Query System for Internet Databases," Proc. ACM SIGMOD.

Cheng, R., Kalashnikov, D.V. and Prabhakar, S. 2004. Querying Imprecise Data in Moving Object Environments," IEEE Trans. *Knowledge and Data Eng.,* 16(9): 1112- 1127.

Chon, H.D., Agrawal, D. and Abbadi, A.E. 2003. Range and Knn Query Processing for Moving Objects in Grid Model," ACM Kluwer MONET, 8: (4)401-412.

Chow, C.Y., Mokbel, M.F. and Liu, X. 2006. A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-Based Services," Proc. ACM Int'l Symp. Geographic Information Systems (GIS), 171-178.

Gedik, B. and Liu, L. 2004. MobiEyes: Distributed Processing of Continuously Moving Queries on Moving Objects in a Mobile System," Proc. Int'l Conf. Extending DataBase Technology (EDBT).

Gedik, B. and Liu, L. 2005. Location Privacy in Mobile Systems: A Personalized Anonymization Model," Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS), pp. 620-629.

Gedik, B. and Liu, L. 2008. Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms," IEEE Trans. Mobile Computing, 7(1): 1-18.

Gedik, B. and Liu, L. 2008. Protecting Location Privacy with Personalized-Anonymity: Architecture and Algorithms," IEEE Trans. Mobile Computing, 7(1): pp. 1-18.

Ghinita, G., Kalnis, P. and Skiadopoulos, S. 2007. Mobihide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries," Proc. Int'l Symp. Spatial and Temporal Databases (SSTD).

Ghinita, G., Kalnis, P. and Skiadopoulos, S. 2007a. Mobihide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries,"Proc. Int'l Symp. Spatial and Temporal Databases (SSTD).

Ghinita, G., Kalnis, P. and Skiadopoulos, S. 2007. Prive: Anonymous Location-Based Queries in Distributed Mobile Systems," Proc. Int'l World Wide Web Conf. (WWW '07), pp. 371-380.

Ghinita, G., Kalnis, P. and Skiadopoulos, S. 2007b. Prive: Anonymous Location-Based Queries in Distributed Mobile Systems," Proc. Int'l World Wide Web Conf. (WWW '07), pp. 371-380.

Gruteser, M. and Grunwald, D. 2003. Anonymous Usage of Location-Based Services through Spatial and Temporal Cloaking," Proc.MobiSys.

Guttman, A. 1984. R-Trees: A Dynamic Index Structure for SpatialSearching," Proc. ACM SIGMOD.

Hjaltason, G.R. and Samet, H. 1999. Distance Browsing in Spatial Databases," ACM Trans. Database Systems, 24(2): 265-318.

Hu, H., Xu, J. and Lee, D.L. 2005. A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," Proc. ACMSIGMOD, pp. 479-490.

Iwerks, G., Samet, H. and Smith, K. 2003. Continuous k-Nearest Neighbor Queries for Continuously Moving Points with Updates. Proc. Int'l Conf. Very Large Data Bases (VLDB).

Iwerks, G.S., Samet, H. and Smith, K. 2004. Maintenance of Spatial Semijoin Queries on Moving Points," Proc. Int'l Conf. Very Large Data Bases (VLDB).

Jensen, C.S., Lin, D. and Ooi, B.C. 2004. Query and Update Efficient B+-Tree Based Indexing of Moving Objects," Proc. Int'l Conf. Very Large Data Bases (VLDB).

Kalashnikov, D.V., Prabhakar, S. and Hambrusch, S.E. 2004. Main Memory Evaluation of Monitoring Queries over Moving Objects,"Distributed Parallel Databases, 15(2): 117-135.

Kalnis, P., Ghinita, G., Mouratidis, K. and Papadias, D. 2007. Preventing Location-Based Identity Inference in Anonymous Spatial Queries," IEEE Trans. Knowledge and Data Eng., 19(12): 1719-1733.

Khoshgozaran, A. and Shahabi, C. 2007. Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. Proc. Int'l Symp. Spatial and Temporal atabasesSSTD.

Pfoser, D. and Jensen, C.S. 1999. Capturing the Uncertainty of Moving- Objects Representations," Proc. Int'l Conf. Scientific and Statistical Database Management (SSDBM).

****