An Approach for Location privacyFramework for Continuously Moving Objects

Madhavi .Macharapu
Department Of Computer Science and Engineering
(M.tech) JNTU UniversityHyderabad,India

Abstract

In location-based services, users with location-aware mobile devices are able to make queries about their surroundings anywhere and at any time. Advances in sensing and tracking technologies create new opportunities for location-based applications but they also create significant privacy risks. According to the report by Computer Science and Telecommunications Board on *ITRoadmap to a Geospatial Future* [2], location based services (LBSs) are expected to form an important part of the future computing environments that will seamlessly and ubiquitously integrate into our life . An important privacy issue in Location Based Services (LBS) is to hide a user's identity while still provide quality location based services. In this paper, we develop a framework for mobile clients to hide their own identities by cloaking their locations. Experimental results show that our proposed mobility-aware cloaking algorithms significantly improve the quality of location cloaking in terms of an entropy measure without compromising much on query latency or communication cost.

Index Terms—spatial databases, location-based Services , mobile applications
.

1 Introduction

Many countries recognize privacy as a right and have attempted to codify it in law. Location-based services provide mobile users personalized services from their current locations using one of several positioning technologies, e.g. GPS, cell phone positioning, and positioning through Wi-Fi access points. Examples of location based services include wireless 911 emergency service, traffic advisories, location-aware advertising, tourist services, location-based games, and navigation. Location information is sensitive because it is ubiquitous
and can lead to much other information. For example, the frequently visited place such as a hospital may release a person's health condition. Through commonly visited places, people can be linked together. The risks of location privacy breach range from releasing information about visits to sensitive places to enabling unwanted virtual or physical stalking. Since positioning touches upon delicate privacy issues (checking where a person is), strict ethics and privacy
measures are strongly recommended for services that use positioning.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. 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.

In this paper we propose 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.

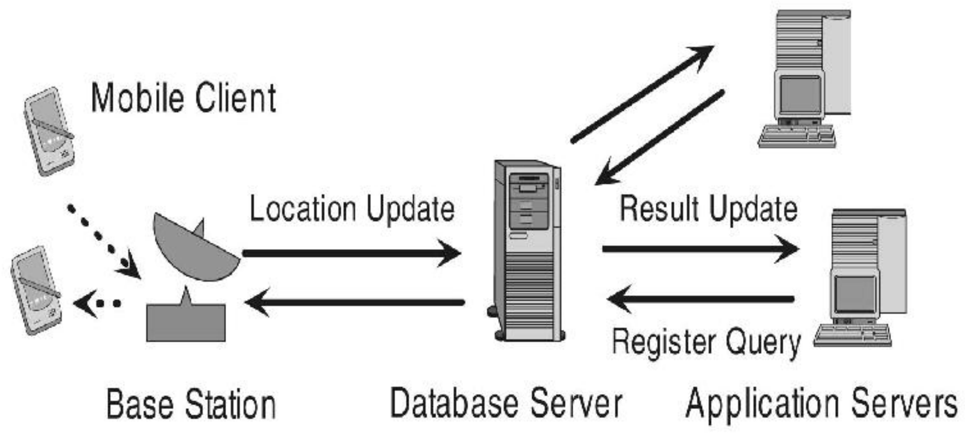


Figure. 1 System architecture

Location cloaking is one typical approach to protecting user location privacy in LBS. Upon receiving a location-based spatial query (e.g., a range query or a kNN query) from the user, the system cloaks the user's current location into a *cloaking region* based on the user's privacy requirement. The location-based spatial query is thus transformed into a *region-based spatial query* before being sent to the LBS server. The LBS server then evaluates the region-based query and returns a *result superset*, which contains the query results for all possible location points in the cloaking region. Finally, the system refines the result superset to generate the exact results for the query location.

In this paper, we identify and address two issues concerning the location cloaking approach. We first show that the representation of a cloaking region has an impact on the result superset size of the region-based query. Second, we consider the location cloaking problem for continuous LBS queries.

2. RELATED WORK

Early work assumed a static data set and focused on efficient access methods (e.g., R-tree) and query evaluation algorithms. Recently, a lot of attention has been paid to moving-object databases, where data objects or queries or both of them move. Similarly, a prototype for interactive cloaking of user locations, based on the PROBE [1mdm] privacy system. PROBE provides privacy guarantees against the association of users with sensitive locations (e.g.,

hospitals, bars, etc). Moreover, privacy can be easily personalized by specifying preferences about the sensitive locations and the desired degree of protection. Based on the user's profile and the map of sensitive locations, PROBE efficiently generates an *obfuscated map* that disguises sensitive locations. Users are presented with a clear and intuitive picture of how their location is protected, improving user experience and increasing privacy awareness.

Previous work has addressed the problem of location K-anonymity either based on centralized or decentralized schemes [2]. However, a centralized scheme relies on an anonymizing server (AS) for location cloaking, which may become the performance bottleneck when there are large More importantly, holding information in a centralized place is more vulnerable to malicious attacks.
A decentralized scheme depends on peer communication to cloak locations and is more scalable. A new hybrid framework called HiSC [2] that balances the load between the (AS)and mobile clients has been proposed. HiSC partitions the space into base cells and a mobile client claims a surrounding area consisting of base cells. The number of mobile clients in the surrounding cells is kept and updated at both client and AS sides. A mobile client can either request cloaking service from the centralized AS or use a peer-to-peer approach for spatial cloaking based on personalized privacy, response time, and service quality requirements.However, it may pose too much computation and communication overhead to the clients.

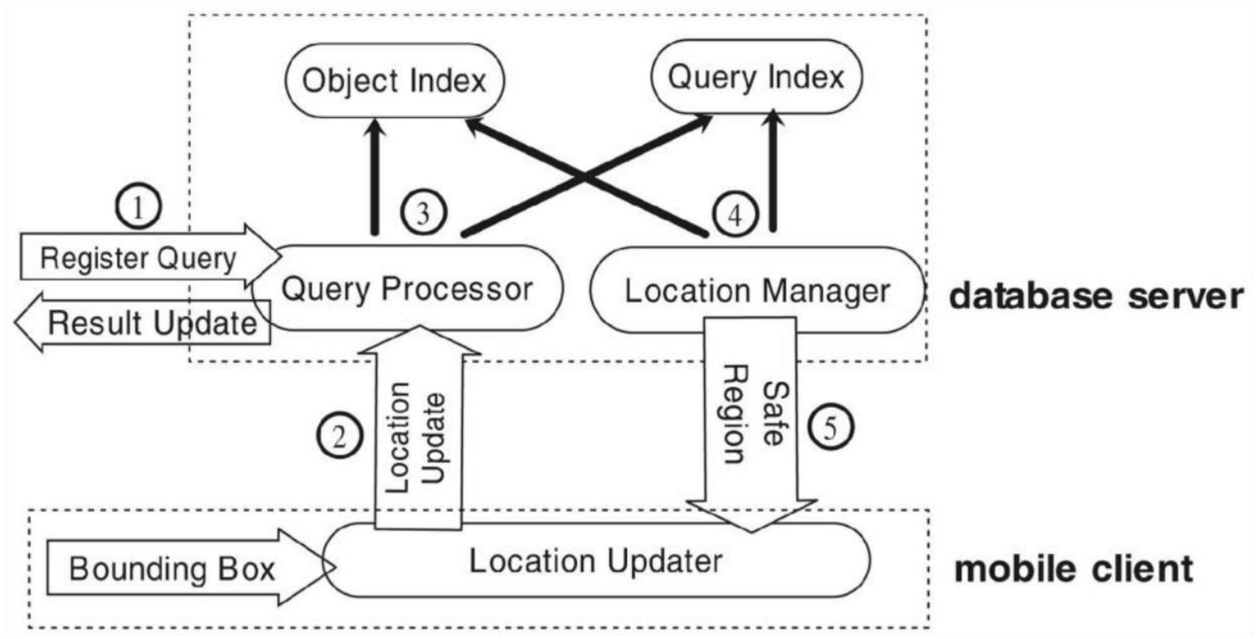
To preserve user location privacy, spatial cloaking is the most commonly used privacy enhancing technique in LBS. The basic idea of the spatial cloaking technique is to blur a user's exact location into a cloaked area that satisfies the user specified privacy requirements. nfortunately, existing spatial cloaking algorithms designed for LBS rely on fixed communication infrastructure, e.g., base stations, and centralized/distributed servers. Thus, these algorithms cannot be applied to a mobile peer-to-peer (P2P) [3] environment where mobile users can only communicate with other peers through P2P multi-hop routing without any support of fixed communication infrastructure or servers.

In terms of architecture models, existing spatial cloaking techniques can be categorized into three models, centralized, distributed and peer-to-peer. For the centralized architecture model [4], [5], a trusted third party, termed location anonymizer, is placed between the user and the service provider. The location anonymizer is responsible for blurring users' exact locations into cloaked areas that satisfy their privacy requirements, and for communicating with the service provider. This architecture model could pose a scalability issue because it requires all the mobile users to periodically report their exact locations to the location anonymizer. Also, storing the user's exact location at a server could pose a privacy breach, i.e., a single point of attacks.
For the distributed architecture model, the users maintain a complex data structure to anonymize their location information through fixed communication infrastructure, i.e., base stations. However, such a complex data structure leads to difficulties to apply this model to highly dynamic location based mobile applications. The basic idea is that mobile users are able to work together to blur their locations into cloaked areas without using any fixed communication infrastructure or centralized/distributed servers.
Distributed approaches have been investigated to monitor continuous range queries [6], and continuous kNN queries. 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, data streams, and sensor databases. However, these studies are not applicable to monitoring of moving objects, where a two-dimensional space is assumed.

2.1 Framework

This 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.
• 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.



This framework works as follows (see Fig. 2): 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.

Algorithm1. Overview of Database Behavior

```
1: while receiving a request do
2: if the request is to register query q then
3: evaluate q;
4: compute its quarantine area and insert it into the query index;
5: return the results to the application server;
6: update the changed safe regions of objects;
7: else if the request is to deregister query q then
8: remove q from the query index;
9: else if the request is a location update from object p then
10: determine the set of affected queries;
11: for each affected query q0 do
12: reevaluate q0;
13: update the results to the application server;
14:  recompute its quarantine area and update the query index;
15: update the safe region of p;
```

2.2 Safe Region

Our basic idea is to maintain a rectangular area, called safe region, for each object. The saferegion 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.

The key idea to solving the problem is "safe region," which was defined in [7] 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.

2.3. Location Cloaking

Location cloaking in the presence of node mobility is more challenging. Right after the network is deployed, each node can find its initial cloaking box using the domain partitioning technique discussed in the previous subsection. One minor change is the minimal safety level that governs when a subdomain should be partitioned further.

Given a partition P, its safety level $S(P)$ downgrades when a node moves into it. Given the required safety level θ, the maximum number of nodes allowed in P is $n_{max} = \left\lfloor \frac{A(P)}{\theta} \right\rfloor$. Thus, to ensure $S(P)$ no less than θ before P is merged with some other partitions, $S(P)$ must be at least $\alpha . \theta$, where α, referred to as P's safety coefficient, is equal to n_{max} =[n_{max}/n_{max}-1]. Therefore, the domain partitioning procedure goes to a subdomain P, only if the safety level of its parent partition is no less than $2\alpha \cdot \theta$.

Algorithm 2 Cloak for mobile ad hoc networks

```
M-Cloak($\theta$)//execute by each node
1: {monitor my movement against current partition $P$}
2: if crossing the boundary of $P$ then
3: //update $P$
4: send packet ($LEAVE$, $seqnum$, $P$)
5: $N(P) \leftarrow N(P) - 1$
6: $s \leftarrow A(P) / N(P)$
7: if $s \geq 2\alpha \cdot \theta$ then
8: wait(T) {wait until LEAVE broadcast is finished}
9: S − Cloak(P, $\theta$) {split P}
10: end if
```

```
11: //find new cloaking box

12: {listen and eavesdrop ADVERTISE packet for $P¢$}
13: send packet ($JOIN$, $seqnum$,$P¢$)
14: $N(P¢) \leftarrow N(P¢) + 1$
15: $s \leftarrow A(P¢) / N(P¢)$
16: if $s \le \alpha \cdot \theta$ then
17: wait(T) {wait until JOIN broadcast is finished}
18: while true do
19: $P¢ \leftarrow$ parent partition of $P¢$
20: calculate safety level of $P¢$ {as the same as is S-Cloak}
21: if $S(P¢) \ge \alpha \cdot \theta$ then
22: set $P¢$ as new cloaking box
23: end if
24: end while
25: end if
26: end if
```

In the above process, a node reveals its location information in packets LEAVE, ADVERTISE, JOIN and MERGE. For each packet of (*LEAVE*, *seqnum*, *P*), the location information in the packet cannot be used to locate the sender because the sender is outside *P*. In addition, an ADVERTISE packet is broadcast in the sender's cloaking box, whose safety level is guaranteed to be no less than θ; a MERGE packet is broadcast in the parent partition of sender's cloaking box, whose safety level is also no less than θ. Comparing with LEAVE, ADVERTISE and MERGE, JOIN packets involve a safety problem. When an adversary eavesdrops a JOIN packet of (*JOIN*, *seqnum*, *P*), it knows the population of *P* increases by one. Thus, if the adversary receives multiple JOIN packets in a short period, it may be able to infer that the safety level of *P* downgrades below θ. To cope with this problem, a node should not send JOIN packet immediately after it eavesdrops an ADVERTISE message. Instead, we let a node wait for a random time period before it sends JOIN packet. During the waiting period, if a node receives another JOIN packet, it waits for another certain period which ensures that the eavesdropped JOIN packet has been broadcast in *P*. As long as it does not hear MERGE packet during the waiting period, it sends its JOIN packet. Otherwise, it extends the waiting time further until the merge process finishes. In this way, the JOIN packets are broadcast sequentially, and from perspective of the adversary, the safety level of *P* must be no less than θ before merge happens. An adversary may launch DOS attacks by inserting fake LEAVE or JOIN packets. Such attacks can be prevented by simply letting nodes recalculate the population of their current cloaking box before each split or merge.

2.4. Analysis

In the mobile ad hoc network, the initialization of cloaking boxes is very similar with the stationary ad hoc network. In the analysis, we focus on the cloaking box update and estimate average number of control packets sent by a mobile node per time unit. Suppose *b* is a cloaking box at the depth *k* in the BP-tree. When a node moves out of *b*, it broadcasts a LEAVE packet within *b*, and the cost is $Cleave = N(b)$. If the safety level of *b* becomes larger than 2α *b* is split, and every node inside *b* will recalculate its cloaking box by broadcasting a PLUS packet in *b*'s child partitions.

The cost *Csplit* is bounded by $1/2\ N2(b) \le Csplit \le N2(b)$. The lower bound denotes the cost when mobile nodes in *b* is uniformly distributed; the upper bound denotes the cost when one of *b*'s child partition is empty. On the other hand, if a mobile node moves into *b*, a JOIN packet is broadcast inside *b* with cost $Cjoin = N(b)$. If the safety level of *b* downgrades lower than θ, every node in *b* broadcasts a MERGE packet within *b*'s parent partition *P*(*b*), the cost of which is $N(b)N(P(b))$. Then all nodes in *P*(*b*) recalculate the safety level of *P*(*b*) by broadcasting PLUS packets, the cost of which is $N2(P(b))$.Thus, the total cost of the merging process is $Cmerge = N(P(b))(N(P(b)) + (N(b)))$. Suppose mobile nodes follow a random walk model, in which at every time unit, a mobile node moves with a randomly picked direction and speed. The time duration that a randomly moving unit may stay in an area can be approximated as an exponential

distribution and the mean staying time is $t = \frac{\pi A}{E[v]L}$, where *A* is the area, *L* is the perimeter of the Aarea, and *E*[*v*] is the average speed of the mobile unit.
discussed above, under uniformly distribution, the cloaking boxes of mobile nodes are at depth *dmax* in the BP-tree. Thus, the average time duration that a mobile host stays inside its cloaking box is ,

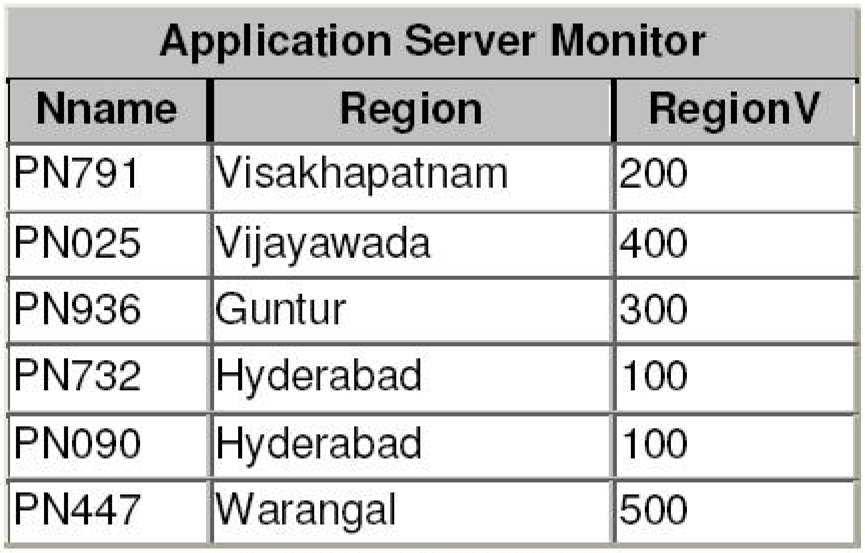
$$\bar{t} = \frac{\pi \cdot l \cdot w}{2^{d_{max}-1} E[v] L(d_{max})}, \text{ where}$$

$$L(k) = \frac{l}{2^{\lfloor k/2 \rfloor - 1}} + \frac{w}{2^{\lfloor (k-1)/2 \rfloor - 1}}$$

where is the perimeter of a partition at depth *k* in the BP-tree. Since the random walk does not change the uniform distribution, theoretically, neither split nor merge happens during the movement of mobile nodes, and the control overhead is composed of only LEAVE and JOIN messages.
In reality, split and merge may happen but the frequency must be very low. Thus, in the analysis, we only count the LEAVE and JOIN messages. Since the cloaking boxes contain only one node, the cost of broadcasting LEAVE and JOIN is equal to 1. Therefore, during cloaking update, the average number of control packets sent by a mobile node per time unit can be estimated as Cupdate = 2/t .

3. Results

Using spatial cloaking technique for mobile adhoc networks, we blur a user's exact location into
a cloaked area that satisfies the user specified privacy requirements. Here RegionV is the cloaked area, Nname is the node name. The RegionV shows 100 until the user is in Hyderabad with in range of 50 and 150 i.e., 100-50 and 100+50. Table.1. Application Server Monitor



Application Server Monitor

Nname	Region	RegionV
PN791	Visakhapatnam	200
PN025	Vijayawada	400
PN936	Guntur	300
PN732	Hyderabad	100
PN090	Hyderabad	100
PN447	Warangal	500

Nname- Node Name
RegionV- Region Coverage

4. CONCLUSIONS

This paper proposes a framework for monitoring continuous spatial queries over randomly moving objects. This framework for mobile clients hides their own identities by cloaking their locations. Experimental results show that our proposed mobility-aware cloaking algorithms significantly improve the quality of location cloaking in terms of an entropy measure without compromising much on query latency or 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

[1] S. Babu and J. Widom, “Continuous Queries over Data Streams,” Proc. ACM SIGMOD, 2001.
[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, “The R*- Tree: An Efficient and Robust Access Method for Points and Rectangles,” Proc. ACM SIGMOD, pp. 322-331, 1990.
[3] R. Benetis, C.S. Jensen, G. Karciauskas, and S. Saltenis, “Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects,” Proc. Int’l Database Eng. and Applications Symp. (IDEAS), 2002.
[4] A. Beresford and F. Stajano, “Location Privacy in Pervasive Computing,” IEEE Pervasive Computing, vol. 2, no. 1, pp. 46-55, Jan.-Mar. 2003.
[5] J. Broch, D.A. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva, “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols,” Proc. ACM/IEEE MobiCom, pp. 85-97, 1998.
[6] Y. Cai, K.A. Hua, and G. Cao, “Processing Range-Monitoring Queries on Heterogeneous Mobile Objects,” Proc. IEEE Int’l Conf. Mobile Data Management (MDM), 2004.
[7] J. Chen and R. Cheng, “Efficient Evaluation of Imprecise Location-Dependent Queries,” Proc. IEEE Int’l Conf. Data Eng. (ICDE), pp. 586-595, 2007.
[8] J. Chen, D. DeWitt, F. Tian, and Y. Wang, “NiagaraCQ: A Scalable Continuous Query System for Internet Databases,” Proc. ACM SIGMOD, 2000.
[9] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, “Querying Imprecise Data in Moving Object Environments,” IEEE Trans. Knowledge and Data Eng., vol. 16, no. 9, pp. 1112- 1127, Sept. 2004. [10] H.D. Chon, D. Agrawal, and A.E. Abbadi, “Range and kNN Query Processing for Moving Objects in Grid Model,” ACM/
Kluwer MONET, vol. 8, no. 4, pp. 401-412, 2003.

[11] C.-Y. Chow, M.F. Mokbel, and X. Liu, "A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-Based Services," Proc. ACM Int'l Symp. Geographic Information Systems (GIS), pp. 171-178, 2006.
[12] D. Pfoser and C.S. Jensen, "Capturing the Uncertainty of Moving- Objects Representations," Proc. Int'l Conf. Scientific and Statistical Database SSDBM), 1999.Management .
[13] B. Gedik and L. Liu, "MobiEyes: Distributed Processing of Continuously Moving Queries on Moving Objects in a Mobile System," Proc. Int'l Conf. Extending DataBase Technology (EDBT), 2004.
[14] B. Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS), pp. 620-629, 2005.
418 IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 3, MARCH 2010
Fig. 22. Minimum cost strategy versus standard strategy. (a) Accuracy. (b) Communication cost.
Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 09,2010 at 09:47:07 UTC from IEEE Xplore. Restrictions apply.
[15] B. Gedik and L. Liu, "Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms," IEEE Trans.

Mobile Computing, vol. 7, no. 1, pp. 1-18, Jan. 2008.
[16] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Mobihide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries," Proc. Int'l Symp. Spatial and Temporal Databases (SSTD), 2007. [17] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Prive: Anonymous Location-Based Queries in Distributed Mobile Systems," Proc. Int'l World Wide Web Conf. (WWW '07), pp. 371-380, 2007.
[18] M. Gruteser and D. Grunwald, "Anonymous Usage of Location- Based Services through Spatial and Temporal Cloaking," Proc. MobiSys, 2003.
[19] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD, 1984.
[20] G.R. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases," ACM Trans. Database Systems, vol. 24, no. 2, pp. 265-318, 1999.
[21] H. Hu, J. Xu, and D.L. Lee, "A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," Proc. ACM SIGMOD, pp. 479-490, 2005.
[22] G. Iwerks, H. Samet, and K. Smith, "Continuous k-Nearest Neighbor Queries for Continuously Moving Points with Updates," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2003.
[23] G.S. Iwerks, H. Samet, and K. Smith, "Maintenance of Spatial Semijoin Queries on Moving Points," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2004.