International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

A Security Model for Protecting Location Based Queries

***N.PRASANNA** ****K.MAHESH**

*M.TECH student , Dept of CSE, VAAGDEVI COLLEGE OF ENGINEERING
**Assistant Professor, Dept of CSE , VAAGDEVI COLLEGE OF ENGINEERING

ABSTRACT: This paper presents a solution to one of the location-based query problems. This problem is defined as follows: (i) a user wants to query a database of location data, known as Points Of Interest (POI), and does not want to reveal his/her location to the server due to privacy concerns; (ii) the owner of the location data, that is, the location server, does not want to simply distribute its data to all users. The location server desires to have some control over its data, since the data is its asset.A location based query solution that employs two protocols that enables a user to privately determine and acquire location data. The first step is for a user to privately determine his/her location using oblivious transfer on a public grid. Oblivious Transfer used to achieve a more secure solution for both parties. The second step involves a private information retrieval interaction that retrieves the record with high communication efficiency. The solution which is present is efficient and practical in many scenarios. This paper includes the results of a working prototype to illustrate the efficiency of our protocol.

KEYWORDS: content based mining, location based services, NN(nearest Neighbour) , POI (points of Interest), Valid region (VR)

I. INTRODUCTION

Location Based Services (LBSs), also known as location dependent information services (LDISs), have been recognized as an important context-aware application in pervasive computing environments. Spatial queries are one of the most important LBSs. According to spatial constraints, spatial queries can be divided into several categories including nearest neighbor (NN) queries and window queries. An NN query is to find the nearest data object with respect to the location at which the query is issued (referred to as the query location of the NN query). For example, a user may launch an NN query like "show the nearest coffee shop with respect to my current location." On the other hand, a window query is to find all the objects within a specific window frame. An example window query is "show all restaurants in my car navigation window." In general, a mobile client continuously launches spatial queries until the client obtains a satisfactory answer. For example, a query "show me the rate of the nearest hotel with respect tomy current location" is continuously submitted in a moving car so as to find a desired hotel. The naive method answering continuous spatial queries is to submit a new query whenever the query location changes. The naïve method is able to provide correct results, but it poses the following problems: High power consumption. The power consumption of a mobile device is high since the mobile device keeps submitting queries to the LBS server. Heavy server load. A continuous query usually consists of a number of queries to the LBS server, thereby increasing the load on the LBS server. Fortunately, in the real world, the queries of a continuous query usually exhibit spatial locality. Thus, caching the query result and the corresponding valid region (VR) in the client side cache was proposed to mitigate the above problems. The valid region, also known as the valid scope, of a query is the region where the answer of the query remains valid. Subsequent queries can be avoided as long as the client is in the valid region. In this paper, we focus on the efficient processing of location dependent queries and, in particular, a sub-class of queries called *mobile nearest-neighbor (NN) search*. A mobile *NN* search is issued by a mobile client to retrieve stationary service objects nearest to its user.1 It is an important function for LBSs, but the implementation is difficult since the clients are mobile and queries must be answered based on the clients' current locations. If a client keeps moving after it issued a query, the query result would continue to change in accordance with the client's movement. As such, it is difficult to obtain results which are accurate with respect to the position pat which the user receives them. Despite the fact that LBSs open up new research opportunities, most of the on-going research work still concentrates on traditional queries which return answers independent to the locations of the query issuers. In other words, each data object has only one set of attribute values in the server. If a client caches a local copy of the data to improve performance, the cached data become invalid only when the corresponding copy in the server is updated. As for location-dependent queries, a data object usually has multiple sets of attribute values, each of which is valid only when the client is located within a specific region. While mobile data caching and invalidation for locationindependent queries has been actively pursued in the mobile computing research community, very few work had been done on indexing and query processing techniques for location-dependent queries.

II. RELATED WORK

The advent of high-speed wireless networks and the popularity of portable devices have fueled the development of mobile computing. Compared to traditional computing paradigms, mobile computing enables clients to have unrestricted mobility while maintaining network connection. The ability of users to move and identify their own locations opens up a new kind of information services, called location-dependent information services (LDISs), which produce the answer to a query according to the location of the client issuing the query . Examples of mobile LDISs include nearest object searching (e.g., finding the nearest restaurant) and local information access (e.g., local traffic, news, and attractions. The spatial property of location-dependent data introduces new problems for data caching research. First, the cached result for a query (e.g., the nearest restaurant) may become invalid when the client moves from one location to another. The maintenance of the validity of the cached data when the client changes location is called location-dependent cache invalidation. Second, the cache replacement policy on the client has to

consider the sizes of the valid scopes p(hereinafter called valid scope areas) of the cached values. The valid scope of a data value is defined as the geographical area within which the data value is valid. When the valid scope of a data value is large, the chance for the client to issue the same query within the valid scope, thus generating a cache hit, is also large. As such, the cache replacement policy should try to retain the data value with pa larger valid scope area in the cache. [1] Owing to increasing demands from mobile users, *Location-Based Services* (LBSs) have received a lot of attention in recent years. Examples of queries for location-based services include "find the nearest gas station from my current location", "find all the cinemas within 1 km radius", "which buses will pass by me in the next 10 minutes?" and so on. While data objects in the first two examples are stationary, those in the last example are mobile. In this paper, we focus on queries issued by mobile users on relatively static data objects, because they are the most common kind of queries in LBSs. The movement of mobile clients presents many new research problems for location-dependent query processing there are several technical issues involved with the implementation of an LBS, which include locating the position of a mobile user, tracking and predicting movements, processing queries efficiently, and bounding location errors. [2] Consider a computing environment with a large number of location-aware mobile objects. We want to retrieve the mobile objects inside a set of user-defined spatial regions and continuously monitor the population of these windows over a time period. In this paper, we refer to such continuous queries as *range-monitoring queries*. Efficient processing of range-monitoring queries could enable many useful applications. similarly, we might want to track traffic condition pin some area and dispatch more police to the region if the number of vehicles inside exceeds a certain threshold. In such applications, it is highly desirable and sometime critical to provide accurate results and update them pin real time whenever mobile objects enter or exit the regions of interest. Unlike conventional range queries, a range-monitoring query is a continuous query. It stays active until it is terminated explicitly by the user. As objects continue to move, the query results change accordingly and require continuous updates. A simple strategy for computing range monitoring queries is to have each object report its position as it moves. The server uses this information to identify the affected queries, and updates their results accordingly. This simple approach requires excessive location updates, and obviously is not scalable. Each location update consists of two expenses – mobile communication cost and server processing cost. If a battery-powered object has to constantly report its location, the battery would be exhausted very quickly. It is well-known that sending a wireless message consumes substantially

more energy than running simple procedures . [3] .

Mobile devices with computational, storage, and wireless communication capabilities (such as PDAs) are becoming increasingly popular. At the same time, the technology behind positioning systems is constantly evolving, enabling the integration of low cost GPS devices in any portable unit. Consequently, new mobile computing applications are expected to emerge, allowing users to issue location-dependent queries in a ubiquitous manner. Consider, for instance, a user (mobile client) in an unfamiliar city, who would like to know the 10 closest restaurants. This is an instance of a *k nearest neighbor* (*k*NN) query, where the query point is the current location of the client and the set of data objects contains the city restaurants. Alternatively, the user may ask for all restaurants located within a certain distance, i.e., within 200 meters.[5] This is an instance of a *range* query. Spatial queries have been studied extensively in the past, and numerous algorithms exist (for processing snapshot queries on static data indexed by a spatial access method. Subsequent methods focused on moving queries (clients) and/or objects. The main idea is to return some additional information (e.g., more NNs expiry time validity region that determines the lifespan of the result. Thus, a moving client needs to issue another query only after the current result expires.

These methods focus on single query processing, make certain assumptions about object movement and do not include mechanisms for maintenance of the query results (i.e., when the result expires, a new query must be issued). Recent research considers *continuous monitoring* of multiple queries over arbitrarily moving objects. In this setting, there is a central server that monitors the locations of both objects and queries. The task of the server is to report and continuously update the query results as the clients and the objects move. As an example, consider that the data objects are vacant cabs and the clients are pedestrians that wish to know their *k* closest free taxis until they hire one. [6]As the reverse case, the queries may correspond to vacant cabs, and each free taxi driver wishes to be continuously informed about his/her *k* closest pedestrians. Several monitoring methods have been proposed, covering both range and *k*NN queries. Some of these methods assume that objects issue updates whenever they move, while others consider that data objects have some computational capabilities, so that they inform the server only when their movement influences some query.

III. LOCATION BASED QUERIES

In this paper, we propose a novel protocol for location based queries that has major performance improvements with respect to the approach by Ghinita. Like such protocol, our protocol is organized according to two stages. In the first stage, the user privately determines his/her location within a public grid, using oblivious transfer associated symmetric key for the block of data in the private grid. In the second stage, the user executes a communicational efficient PIR, to retrieve the appropriate block in the private grid. This block is decrypted using the symmetric key obtained in the previous stage. Our protocol thus provides protection for both the user and the server. The user is protected because the server is unable to determine his/her location. Similarly, the server's data is protected since a malicious user can only decrypt the block of data obtained by PIR with the encryption key acquired in the previous stage. In other words, users cannot gain any more data than what they have paid for. We also provide results from a working prototype showing the efficiency of our approach.

3.1 SYSTEM DESIGN

The proposed system architecture for NN and window query processing. The system architecture consists of three parts: 1) an external LBS server, 2) deployed proxies, and 3) the mobile clients. The LBS server is responsible for managing static data objects and answering the queries submitted by the proxies. Note that the LBS server can use any index structure (e.g., R-tree or grid index) to process spatial queries. The LBS server is assumed not to provide VRs. Each of the deployed proxies supervises one service area and provides of window queries for mobile clients in the service area. Each base station serves as an intermediate relay for queries and query results between mobile clients and the associated proxy. Base stations, proxies, and the LBS server are connected by a wired network. A mobile client maintains a cache to store the query results and the corresponding. When a mobile client has a spatial query, the mobile device first examines whether the current location is in the of the stored result. If so, the stored result remains valid and the mobile device directly shows it to the client. Otherwise, the mobile device submits the query, which is received and then forwarded by the base station, to the proxy. For the received query, the proxy will return the query result as well as the corresponding EVR to the client

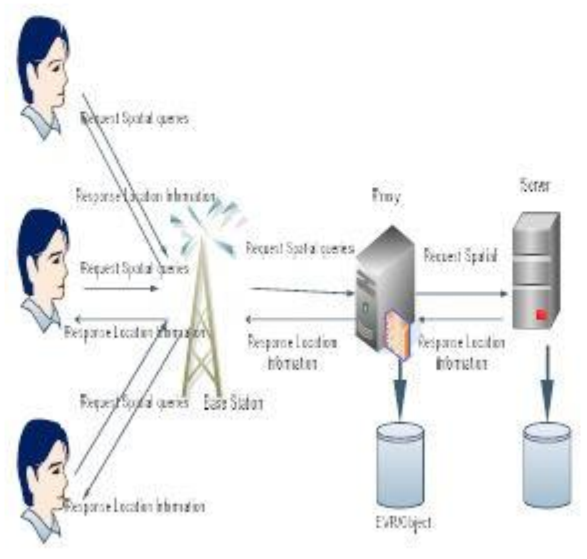

Request Spatial queries
Response Location Information
Request Spatial queries
Response Location Information
Request Spatial queries
Response Location Information
Base Station
Request Spatial queries
Response Location Information
Proxy
Request Spatial
Response Location Information
Server
EVRObject

Fig.1Overall system design

3.2.1 Clients Process

The mobile device submits the query, which is received and then forwarded by the base station, to the proxy. For the received query, the proxy will return the query result as well as the corresponding EVR to the client.The mobile device submits the number of query to proxy.A large number of queries submitted by mobile clients.

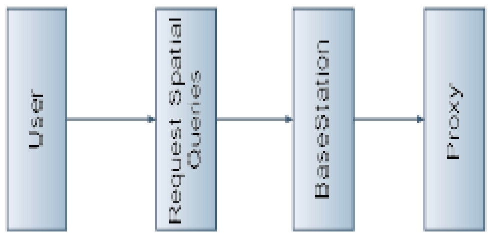


Fig.2 Client process flow

3.2.2 Proxy Module:

In this module build proxy server to estimate EVRs of NN queries and EWVs of window queries based on NN query history and available data objects. The LBS server can use any index structure e.g., R-tree or grid index to process spatial queries. The proxy maintains an object cache and two index structures: an EVR-tree for NN queries and a grid index for window queries. The two index structures share the data objects in the object cache.

EVR-TREE GENERATION FOR NN

The EVR-tree is an R-tree (or its variants) composed of EVRs where each EVR is wrapped in a minimum bounding box (MBR). An EVR consists of the region vertices with respect to a data object and a pointer to the corresponding object entry in the object cache. When an NN query point q is located in an EVR of the EVR-tree, the proxy retrievesthe corresponding object from the object cache to answer the query.

GENERATION OF GRID CELL FOR WINDOW QUERIES

Grid cells are classified into two categories: fully cached cells and uncached cells All grid cells are initialized to uncached. The proxy marks a cell as fully cached when all the objects within the cell are received. The corresponding grid index entry of a fully cached cell caches the object pointers to the associated object entries in the object cache. The purpose of fully cached and uncached cells is to realize the stored object distribution, enabling the proxy to create EWVs of window queries effectively. When receiving a window query, the proxy obtains the result and creates the corresponding EWV by retrieving stored objects in the surrounding fully cached cells

3.2.3. Server Module

The LBS server is responsible for managing static data objects and answering the queries submitted by the proxies.

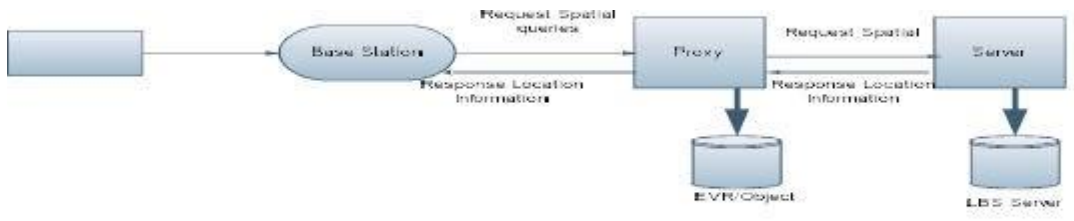


Fig.3Flow of Server and proxy module

3.2.4 Sharing-based nearest neighbor query Module

The sharing-based nearest neighbor query Module provides a rendering of the verification process of a sharing-based NN query in a step-by-step manner. Users can arbitrarily select a mobile host and launch a Locationbased NN query within region.

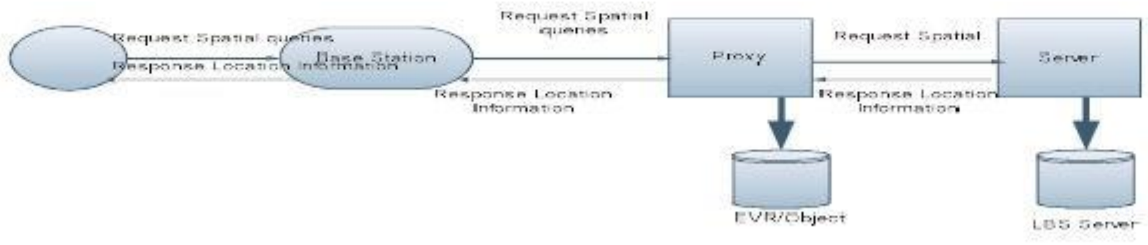


Fig.4Flow of Nearest neighbour query module

3.3 ALGORITHM

R Tree algorithm:

R-trees can be more efficient for data storage and speed at search execution time, though they are generally tied to the internal structure of a given data storage system. R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such

as geographical coordinates, rectangles or polygons. A common real-world usage for an R-tree might be to store spatial objects such as restaurant locations or the polygons that typical maps are made of: streets, buildings, outlines of lakes, coastlines,

Grid index algorithm:

The individual cells of a grid system can also be useful as units of aggregation, for example as a precursor to data analysis, presentation, mapping, etc. A grid index is a used for spatial indexing purposes. A wide variety of such grids have been proposed or are currently in use, including grids based on "square" or "rectangular" cells, triangular grids or meshes, hexagonal grids, grids based on diamond-shaped cells, and possibly more. The range is broad and the possibilities are expanding.

Melkman's algorithm:

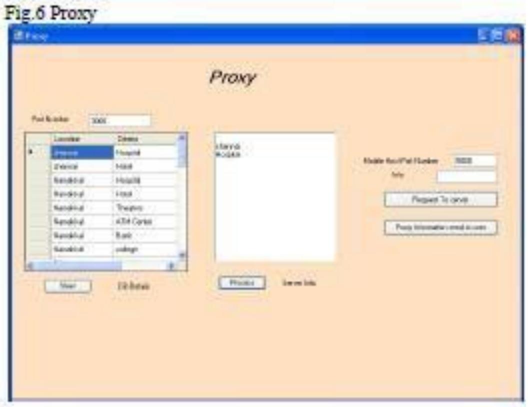
The Melkman's algorithm to compute the convex polygon of the updated EVR to remove the unnecessary vertices and achieve a larger region size. The convex polygon serves as the final updated EVR .

IV. EXPERIMENT SETUP & RESULT

This paper was implemented in .net framwork and backend used sql.

Fig. 5User Send Query

Fig.6 Proxy



http://www.southindia.com/chennai-hospital.html

OK

Fig.7 Here Proxy Receive query from user. It send spatial information and will be Estimate Valid Regions for user.

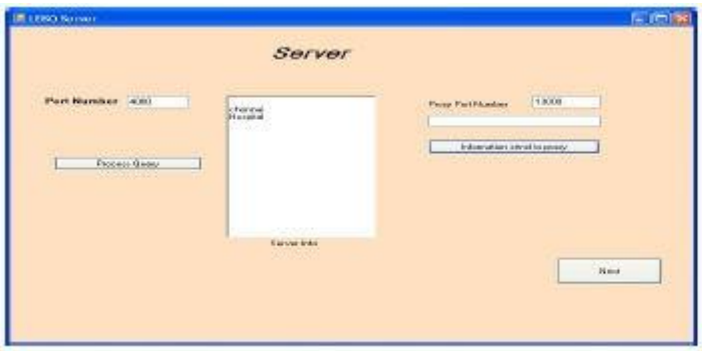


Fig.8 Server UI

LB server Receive query from Proxy. It sends spatial information for Proxy.

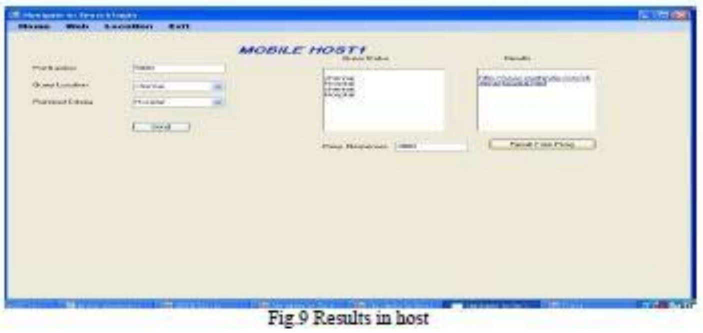


Fig.9 Results in host

User1 receive spatial information from Proxy.

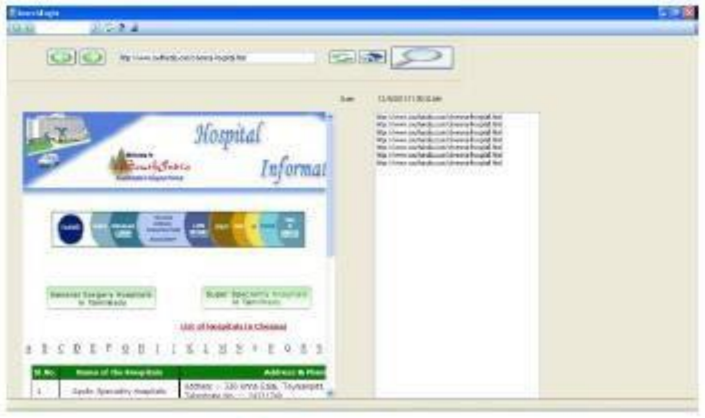


Fig.10 Results in host

V. CONCLUSION

In this paper we have presented a location based query solution that employs two protocols that enables a user to privately determine and acquire location data. The first step is for a user to privately determine his/her location using oblivious transfer on a public grid. The second step involves a private information retrieval inter action that retrieves the record with high communication efficiency. We analyzed the performance of our protocol and found it to be both computationally and communicational more efficient than the solution by Ghinita *et al.*, which is the most recent solution. We implemented a software prototype using a desktop machine and a mobile device. The software prototype demonstrates that our protocol is within practical limits. Future work will involve testing the protocol on many different mobile devices. The mobile result we provide may be different than other mobile devices and software environments. Also, we need to reduce the overhead of the primarily test used in the private information retrieval based protocol. Additionally, the problem concerning the LS supplying misleading data to the client is also interesting. Privacy preserving reputation techniques seem a suitable approach to address such problem. A possible solution could integrate methods from. Once suitable strong solutions exist for the general case, they can be easily integrated into our approach.

REFERENCES

[1] D. Lee, B. Zheng, and W.-C. Lee, "Data Management in Location- Dependent Information Services," IEEE Pervasive Computing, vol. 1, no. 3, pp. 65-72, July-Sept. 2002.

[2] B. Zheng, J. Xu, and D.L. Lee, "Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments," IEEE Trans. Computers, vol. 15, no. 10, pp. 1141-
1153, Oct. 2002.

[3] B. Zheng and D.L. Lee, "Processing Location-Dependent Queries in a Multi-Cell Wireless Environment," Proc. Second ACM Int'l Workshop Data Eng. for Wireless and Mobile Access, 2001.

[4] B. Zheng, J. Xu, W.-C. Lee, and D.L. Lee, "On Semantic Caching and Query Scheduling for Mobile Nearest-Neighbor Search," Wireless Networks, vol. 10, no. 6, pp. 653-664, Dec. 2004.

[5] X. Gao and A. Hurson, "Location Dependent Query Proxy," Proc.ACM Int'l Symp. Applied Computing, pp. 1120-1124, 2005.

[6] X. Gao, J. Sustersic, and A.R. Hurson, "Window Query Processingwith Proxy Cache,"

AUTHOR 1 :-

*N. Prasanna completed her B tech in Vaagdevi College of Engineering and pursuing M-Tech in Vaagdevi College of Engineering

AUTHOR 2:-

**K. Mahesh is working as Assistant Professor in Dept of CSE , Vaagdevi College of Engineering