Global Journal of Computer Science and Technology
Volume 12 Issue 8 Version 1.0 April 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Creating Safe Region for Continuous Moving Objects

By M.Swapna, S.China Venkateswarlu, B.Venkateswarlu & M.Suresh Babu

Holy Mary Institute of Technology/ Affiliated to JNTUH, Hyderabad

Abstract - Monitoring the continuous moving objects while monitoring the two fundamental things are privacy and efficiency are considered. First, evaluating the safe region is nothing but creating the safe region and the monitoring is done when objects are moving from one region to another region. We provide detailed algorithms for query evaluation/reevaluation and safe region computation in this framework. 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, particularly, when and how mobile clients should send location updates to the server. Based on the notions of safe region and most probable result, this project performs location updates only when they would likely alter the query results.

Keywords : *Customer, database monitoring, deregister, object Indexes, Mobile Client, Evaluation, Spatial Query.*

GJCST Classification: *H.2.0*

Strictly as per the compliance and regulations of:

Creating Safe Region for Continuous Moving Objects

M.Swapna [α], S.China Venkateswarlu [σ], B.Venkateswarlu [ρ] & M.Suresh Babu [ω]

Abstract - Monitoring the continuous moving objects while monitoring the two fundamental things are privacy and efficiency are considered. First, evaluating the safe region is nothing but creating the safe region and the monitoring is done when objects are moving from one region to another region. We provide detailed algorithms for query evaluation/reevaluation and safe region computation in this framework. 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, particularly, when and how mobile clients should send location updates to the server. Based on the notions of safe region and most probable result, this project performs location updates only when they would likely alter the query results.

Keywords : Customer, database monitoring, deregister, object Indexes, Mobile Client, Evaluation, Spatial Query.

I. Introduction

This paper is a location based service and it is used to identify the location of the customer. Early work assumed a static data set and focused on efficient access methods 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 in a safe region.

A database monitoring continuous spatial queries over moving objects is needed in numerous applications such as public transportation, logistics and location-based services. In the monitoring system, this consists of a base station, a database server, application servers, and a large number of moving objects like 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. 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 issues like accuracy, efficiency and privacy.

Author α : MTech Student from Dept of CSE-HITS COE, Hyderabad.
Author σ : Professor in Department of ECE ,HITS COE-HYDERABAD.
E-mail : cvenkateswarlus@gmail.com
Author ρ : Assistant Professor in Dept. of CSE-HITS COE,Hyderabad.
Author ω : Assistant Professor in Dept of CSE-MRGI, Hyderabad.

II. Design of Safe region Evaluation & Object Indexs

There are four module, they are as follows

a) Safe Region Evaluation

In this safe region is assumed as a rectangle change of object inside the rectangle would not affect spatial query in the database. 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.

The Client updates its location on the server only when the client moves out of its safe region based on the location of client. The safe region ring is based on the rectangle of the centric.

b) Object Index

Object index is the server side information about spatial query range and used to evaluate safe region.

The object index is the server-side view on all objects. To evaluate queries, the server must store the spatial range, in the form of a bounding box, within which each object can possibly locate. Note that this bounding box is different from a δ- square because its shape also depends on the client-side location updater. That is, it must be a function (denoted by $\odot$) of the last updated δ- square and the safe region. As such, this box is called a bbox as a mark of distinction. In particular, for the standard update strategy, the bbox is the safe region enlarged by $\delta/2$ on each side, or formally, the "Minkowski sum"2 of the safe region and a $\delta/2$- square.

c) The Query Index

Query Index as the following parameter query point, current query result and the quarantine area. The quarantine area is used to identify the queries whose results might be affected by an incoming location update.

For each registered query, the database server stores: 1) the query parameters (e.g., the rectangle of a range query, the query point, and the k value of a kNN query); 2) the current query results; and 3) the quarantine area of the query. The quarantine area is used to identify the queries whose results might be

affected by an incoming location update. It originates from the quarantine line, which is a line that splits the entire space into two regions: the inner region and the outer region. An object becomes a result object if it enters the inner region; likewise, it becomes a non result object once it enters the outer region. However, the ideal quarantine line is difficult to compute, especially in the context of the most probable result. In addition, as object locations have extensions rather than points, the quarantine line is not unique for a query. As such, we allow fuzziness by relaxing the line to an area called "quarantine area." That is, the entire space is split into three regions: the inner region, the quarantine area, and the outer region. The former two are separated by the inner bound of the quarantine area, whereas the latter two are separated by the outer bound of the quarantine area. To ease the computation of these two bounds, an object becomes a result object if its δ- square moves totally inside the inner bound; on the other hand, an object becomes a non result object once its δ- square crosses or is outside the outer bound. Therefore, a query Q is not affected only if "of the updated δ- square p and its last updated δ- square p_{lst}, both of them are totally inside the inner bound or both of them cross or are outside the outer bound of the quarantine area."

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

d) Query Processing

In the PAM framework, based on the object index, the query processor evaluates the most probable result when a new query is registered, or reevaluates the most probable result when a query is affected by location updates.

Obviously, the reevaluation is more efficient as it can be based on previous results.

e) Location Update

Each time client detects the genuine point location, it is wrapped into a bounding box. Then, the client-side location updater decides whether or not to update that box to the server 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.

III. Algorithms

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
5. query index;
6. return the results to the application server;
7. update the safe regions of objects;
8. else if the request is to deregister query q then
9. remove q from the query index;
10. else if the request is a location update from object p
11. then determine the set of affected queries;
12. for each affected query q do
13. reevaluate q;
14. update the results to the application server;
15. recomputed its quarantine area and update the query index;
16. update the safe region of p ;

Algorithm2: Evaluating a new kNN Query
Input: root: root node of object index
Q : the query point
Output: C: the set of KNNs
Procedure

1. Initialize queue H and H;
2. enqueue (root, d(q ,root)} into H;
3. While |C| < k and H is not empty do
4. U= H.pop ();
5. If u is a leaf entry then
6. While d (q,u) >D(q,v) do
7. V=H.pop ();
8. Insert v into H;

Algorithm3: Reevaluating a kNN Query
Input: **C:** Existing set of KNNs
P: The updating object
Output: C: The new set of KNNs
Procedure

1. Initialize queue H and H;
2. Enqueue{ root ,d(q,root)} into H;
3. While d(q,u) >D(q,v) do
4. V=H.pop ();
5. Insert v into H;
6. Else if u is an index entry then
7. For each child entry v into u do
8. Enqueue (v,d(v,q)) into H;
9. Else if u is an index entry then
10. For each child entry v into u do
11. enqueue (v,d(v,q)) into H;

IV. Context Diagram of Work-The System Architecture

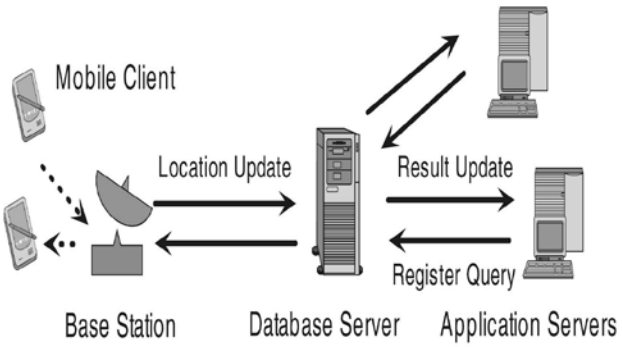


Figure 1: The system architecture

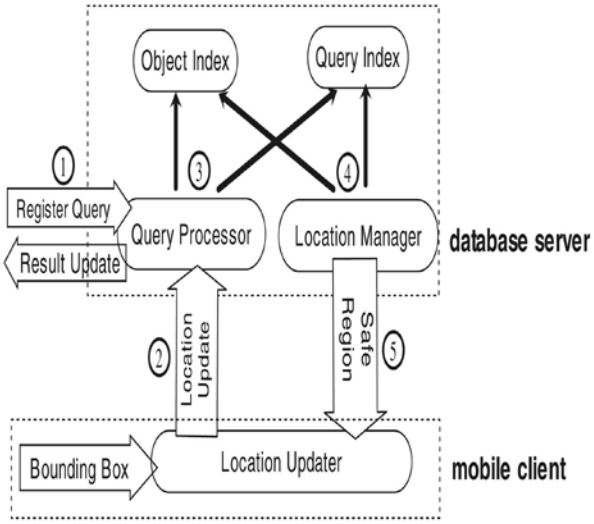
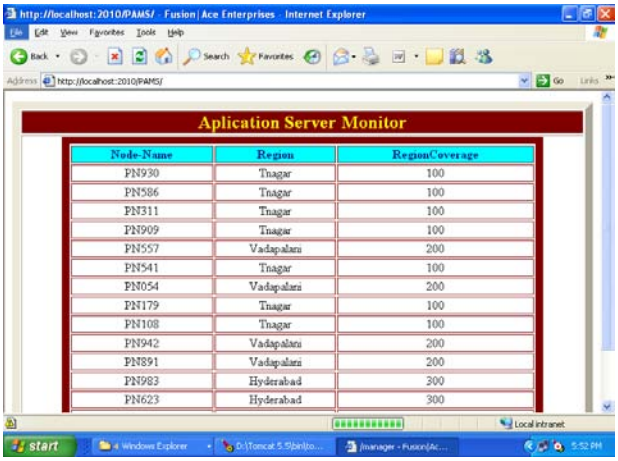


Figure 2 : Framework overview

V. Implementations



Node-Name	Region	RegionCoverage
PN930	Tnagar	100
PN586	Tnagar	100
PN311	Tnagar	100
PN909	Tnagar	100
PN557	Vadapalani	200
PN541	Tnagar	100
PN054	Vadapalani	200
PN179	Tnagar	100
PN108	Tnagar	100
PN942	Vadapalani	200
PN891	Vadapalani	200
PN983	Hyderabad	300
PN623	Hyderabad	300

Figure 3 : Application Server Monitor

This screen will display the monitoring of the application server when the server is on. It contain the node number, region name and distance of that particular region.

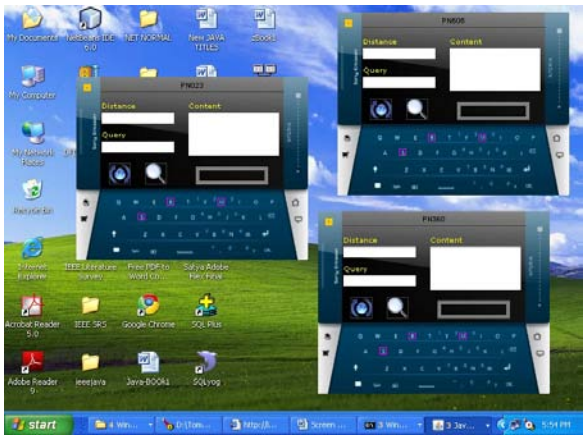


Figure 4 : Creation of multiple nodes

This screen shot is about the opening of the nodes it is done by running few nodes are open.

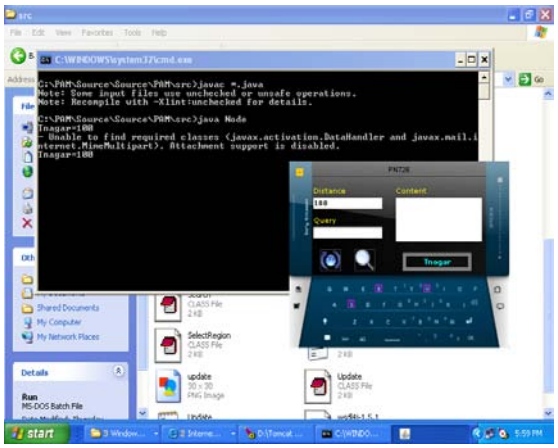


Figure 5 : Entry the query and distance in the node

This screen shot is about run the node and monitoring the server the server is activated for this particular node. By entering the distance it specify whether it is a safe region or not, if it a Safe region it display that particular region otherwise not i.e no region is mentioned.

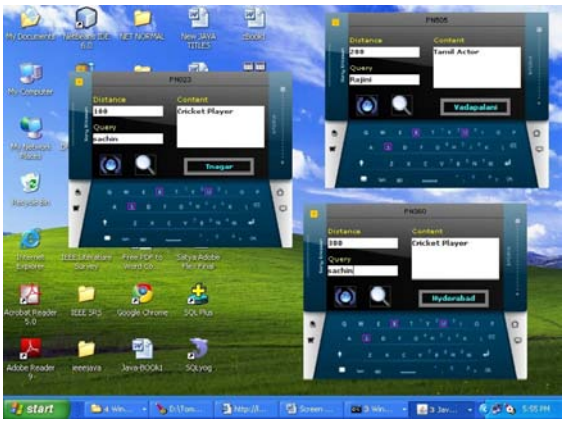
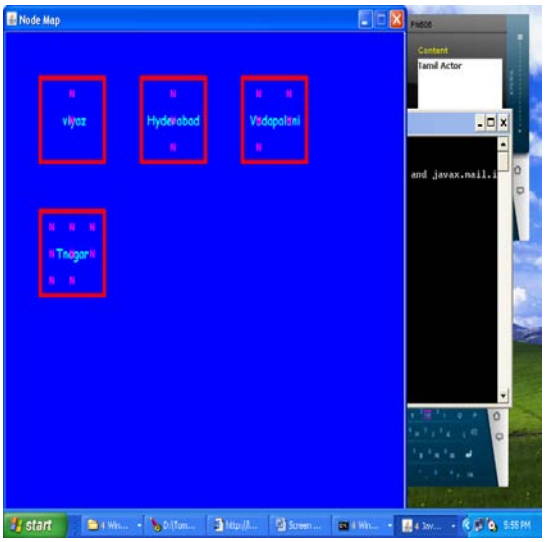


Figure 6 : Displaying the entry of distance and query in the Multiple nodes

This screen shot is about the opening the multiple node by running it. After opening enter the distance after entering the distance it check whether the distance is a safe region or not. If it is a safe region then enter the query, if it also registered query then displays the content otherwise not. This screen shot is a node map it contains few nodes and these nodes are registered node only displayed in the node map. It also contains the queries in that node and these queries are registered queries only. This screen shot is about updated Application server monitored after added the few nodes in the node map.



Screen 7: Displaying the nodes in the node map

VI. Results Analysis

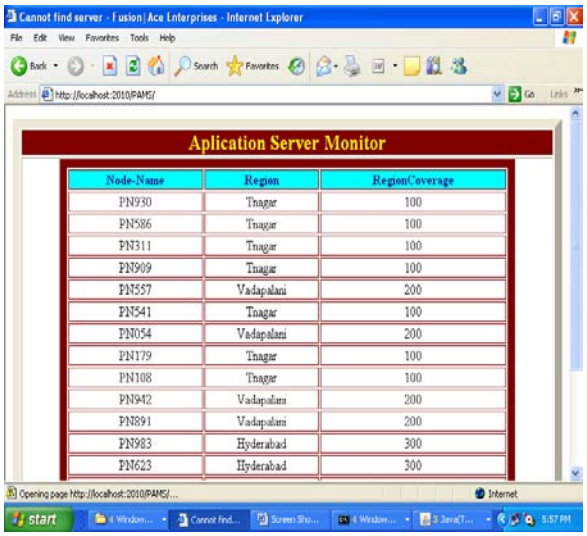


Figure 8 : Update Application Server Monitor

VII. Design of Test Cases

A specific executable test that examines all aspects including inputs and outputs of a system and then provides a detailed description of the steps that should be taken, the results that should be achieved, and other elements that should be identified. Steps explained in a test case include all details even if it they are assumed to be common knowledge. Test cases are used as a technical explanation and reference guide for systems.

Test Case No	Description	Expected Result	Pass/ Fail	Result
1	Enter the distance then click on submit button	It shows the region	pass	It has shown the exact region
2	Enter the query then click on the submit button	It shows the designation of that query	pass	It shown the designation of that query
3	Enter the distance beyond the limit specified in database region then click on submit button	It shows the given distance does not belongs to safe region	pass	It shown the given distance does not belongs to safe region
4	Enter the distance and query then click on submit button	It shows the exact location of the query	pass	It shown the exact location of the query
5	Enter the new distance and query then click on submit button	It shows the current location of the query	pass	It shown the current location of the query

VIII. Conclusion & Future Scope

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. We provide detailed algorithms for query evaluation/ revaluation and safe region computation in this frame-work. We also devise

three-client update strategies that optimize accuracy, privacy, and efficiency respectively. 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. To further optimize the performance of the framework. The minimum cost update strategy shows that the safe region is a crude approximation of the ideal safe area, mainly because of 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. Furthermore, the framework is must be robust and scales well with various parameter settings, such as privacy requirement, moving speed, and the number of queries and moving objects.

Global Journal of Computer Science and Technology Volume XII Issue VIII Version I April 2012
24

This page is intentionally left blank