Bulletin of the Technical Committee on

Data Engineering

September 2005 Vol. 28 No. 3

IEEE Computer Society

Letters

Special Issue on Database Issues for Location Data Management

Conference and Journal Notices

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems.

There are two Data Engineering Bulletin web sites: http://www.research.microsoft.com/research/db/debull and http://sites.computer.org/debull/.
The TC on Data Engineering web page is http://www.ipsi.fraunhofer.de/tcde/.

Letter from the Editor-in-Chief

The Data Engineering Conference ICDE'06

The next International Conference on Data Engineering (ICDE'06) will be held in Atlanta in April, 2006. This conference is the flagship conference of the IEEE Technical Committee on Data Engineering. Atlanta is a great venue, and April is a wonderful time to visit the city, with balmy weather and with magnolias and peach trees in bloom. This year's conference is sure to be of very high quality as submissions continue to be very very high. The result is a very selective conference with high quality papers. Please look for more information about ICDE'06 in the next issue, where additional information about the conference will be available and when a visit to the web site can provide a preview of the technical program.

The Current Issue

When the database field first started up, THE problem to solve was business data processing. That kept the database community fruitfully employed for around 20 years. Over that course of time, business data processing evolved to include not just OLTP and payroll, but decision support, OLAP, data warehouses, data mining, etc. These new directions in business data processing truly kept the database field on its toes over the years.

More recently it has become harder to view databases as only a solution for business data processing. While this area continues to evolve, databases have been used to help tackle other areas as well. The interesting thing about the current issue is that techniques that have attacked these other areas also have overlap with the newer requirements of business data processing. So even as we as a community go off in new directions, many of the things that we do continue to be relevant to the original application area that gave our field its start.

This preamble has surely given the game away for what comes next. Location data management is one of the newer areas of database research. And it has applications in new application areas (mobil phone services come to mind). But the technology also can be turned around and applied to traditional business data processing for inventory control, i.e. you use the technology to keep track of where your inventory is. This kind of application is particularly relevant as just-in-time inventory management increasingly becomes the normal mode of operation.

So location data management looks to have a lively future, not just as a research area, but also as an area with real-life application. It is this kind of real-life relevance that has long distinguished the database field. So I was pleased when Jignesh Patel, this issue's editor, suggested location data management as a topic. Jignesh has pulled together an issue that includes both academic and industrial efforts in this area, demonstrating the wide interest generated in this area. I want to thank Jignesh for the fine job he did on the issue, which should stimulate even more interest in this area.

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

Over the last decade we have witnessed a convergence of various technological forces that have produced a wide-range of location-tracking technologies. Examples of such technologies include the use of Global Positioning System (GPS) devices and Radio Frequency Identification (RFID) technology to tag and track physical objects as they move around in physical space. Rapid advances in semiconductor technologies and efficiencies achieved from producing these location-tracking devices in large volumes have made it possible to manufacture these devices for a small cost, making mass deployments of such devices possible. These location-tracking technologies are enabling a new class of location-based applications that require managing traditional databases and dynamically changing location information. Examples of such location-based application include using location information in vehicle navigation and emergency response, the use of RFID tags for tracking objects in retail supply-chain systems, and the use of location-based information in mobile e-commerce. Managing such location-information poses a number of new challenges for database management systems. The focus of this issue is on the database methods and applications related to location data management.

The first article by Mokbel and Aref presents the PLACE database server which treats the sequence of continually changing location information from a moving object as a spatio-temporal data stream. The PLACE system combines data stream management methods and spatio-temporal query processing methods to produce an efficient database server for handling location-based queries and triggers.

The second article by Cao, Wolfson, Xu, and Yin, presents the MOBI-DIC platform which combines peer-to-peer methods and spatio-temporal data management. Mobile object in their system collaborate to allow searching for local information from mobile devices and also enables the discovery of spatio-temporal resources (such as the currently available free parking slots).

Sending location information from a mobile device to a backend server can consume valuable (and limited) power resources on the mobile device. The third article by Hu and Lee presents a power efficient scheme for updating the location of moving objects.

Recognizing the commercial importance of supporting location-based query processing in the DBMS, Oracle already provides various spatial expressions and rules for supporting location-based services. The next article by Yalamanchi, Kothuri, and Ravada, presents these features which are available in Oracle 10g.

The final article by Krumm and Shafer lays out various practical issues with representing, generating, and efficiently querying location information. This article presents the data management issue for supporting location-based services based on a number of actual examples of such applications.

Location-based applications and notification services are likely to revolutionize the way in which we track physical objects and use dynamic location-based information as we move around in physical space. The five articles presented in this issue provide an overview of some of the challenges and the opportunities in expanding the scope of database management systems to include management of location information. Location-based technologies are predicted to continue it tremendous growth for the next decade and are likely to become even more pervasive in the future. In other words, this area presents significant research and commercial opportunities for the database community. Hopefully, this issue will continue to feed existing interest, and perhaps stimulate new interest, in the database community for this emerging class of application.

Jignesh M. Patel
University of Michigan
Ann Arbor, MI

PLACE: A Scalable Location-aware Database Server for Spatio-temporal Data Streams

Mohamed F. Mokbel
Department of Computer Science and Engineering, University of Minnesota
Minneapolis, MN, 55455
mokbel@cs.umn.edu

Walid G. Aref
Department of Computer Science
Purdue University
West Lafayette, IN 47907
aref@cs.purdue.edu

Abstract

*In this paper, we overview the PLACE server (***P***ervasive* **L***ocation-***A***ware* **C***omputing* **E***nvironments); a scalable location-aware database server developed at Purdue University. The PLACE server extends data streaming management systems to support location-aware environments. Location-aware environments are characterized by the large number of continuous spatio-temporal queries and the infinite nature of spatio-temporal data streams. The PLACE server employs spatio-temporal query operators that support a wide variety of continuous spatio-temporal queries. In addition, the PLACE server is equipped with scalable operators that provide shared execution among multiple continuous spatio-temporal queries. To cope with intervals of high workload of data objects and/or continuous queries, the PLACE server utilizes object and query load shedding techniques to support a larger number of continuous queries with approximate answers.*

1 Introduction

The wide spread of *location-detection* devices (e.g., GPS-like devices, RFIDs, handheld devices, and cellular phones) results in *location-aware* environments where massive spatio-temporal data streams are continuously sent from these devices to database servers. Location-aware environments are characterized by the large numbers of continuously moving objects and moving queries (also known as spatio-temporal queries). Such environments call for new scalable database servers that deal with the continuous movement and frequent updates of both spatio-temporal objects and spatio-temporal queries.

In this paper, we overview the PLACE server (**P**ervasive **L**ocation-**A**ware **C**omputing **E**nvironments) [AHP03, MXA$^+$04b, MXHA04]; a scalable location-aware database server developed at Purdue University. The PLACE server combines the advanced technologies of spatio-temporal databases and data stream management systems to support efficient execution of a large number of continuous spatio-temporal queries over spatio-temporal data streams. The PLACE server is equipped with spatio-temporal pipelined query operators that interact with traditional query operators (e.g., join, distinct, and aggregates) to support a wide variety of continuous spatio-temporal queries. Furthermore, the PLACE server employs *incremental evaluation*, *shared execution*, and *load shedding* techniques to support large number of concurrent spatio-temporal queries.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

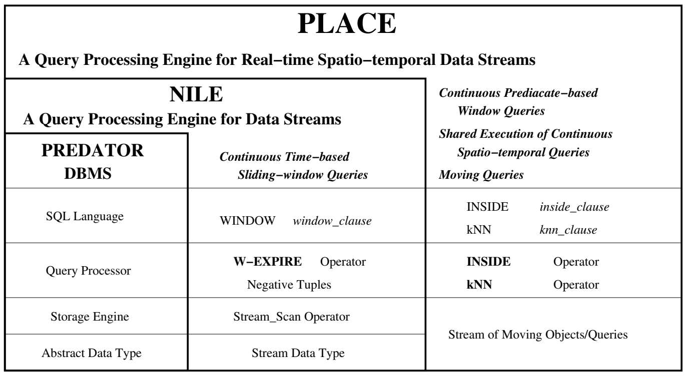


Figure 1: The PLACE Server.

As given in Figure 1, the PLACE server extends the NILE data stream management system [HMA$^+$04] and the PREDATOR database management system [Ses98] to support: (1) Continuous spatio-temporal queries over spatio-temporal data streams (i.e., streams of moving objects) [MXHA04]. (2) The concept of *predicate-based* window queries [GAE05]. (3) Incremental evaluation of continuous spatio-temporal queries [MXA04a]. Incremental evaluation is achieved through updating the query answer by *positive* and *negative* updates. A positive/negative update indicates that a certain object needs to be added to/removed from the query answer. (4) Scalable execution of a large number of continuous spatio-temporal queries [MA05b, MXA04a]. (5) Load shedding techniques for spatio-temporal data streams [MA05b].

The rest of this paper is organized as follows. Section 2 discusses the challenges of dealing with spatio-temporal streams. The spatio-temporal pipeline query operators are presented in Section 3. Sections 4 and 5 give a brief highlight of the *shared execution* and *load shedding* features of the PLACE server, respectively. Finally, Section 6 concludes the paper.

2 Spatio-temporal Data Streams

Although there are numerous research efforts that deal with data streams (e.g., see [GO03] for a survey), the spatial and temporal properties of data streams are addressed only recently in [CM03, HS04] to solve geometric problems (e.g., computing the convex hull) and in [EMA05, TPZL05] to develop spatio-temporal histograms. On the other side, research efforts in spatio-temporal databases (e.g., [CEP03, MGA03, PCC04]) rely mainly on indexing and/or storing the incoming data in disk storage, which is not suitable for the streaming environment. Up to the authors' knowledge, the PLACE server provides the first attempt to furnish query processors in data stream management systems to support continuous queries over spatio-temporal data streams.

2.1 Managing the Scarce Memory

With the infinite nature of spatio-temporal data streams, it becomes essential to develop in-memory algorithms and data structures to support the efficient execution of continuous queries. The PLACE server optimizes the scarce memory resource by keeping track of only those objects that are considered *"significant"*. A moving object P is considered *significant* if there is at least one outstanding continuous query Q that shows interest in P. Thus, once a new incoming data object P_{new} is received, we go through all the outstanding continuous queries to check if there is any query that is interested in P_{new}. If no query shows interest, we ignore the arrival of P_{new}. Moreover, due to the continuous movements of both objects and queries, we monitor the status of all

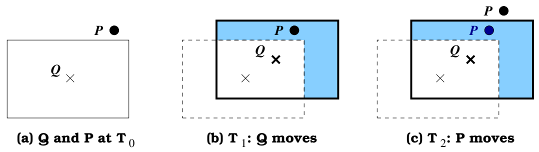


Figure 2: Uncertainty in moving queries.

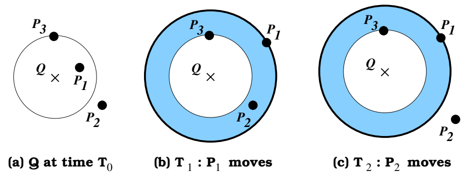


Figure 3: Uncertainty in static NN queries.

the stored objects. If a *significant* stored object becomes *insignificant* at any time, we immediately drop it from memory.

2.2 Uncertainty in Continuous Spatio-temporal Queries

Storing only *significant* objects may result in having *uncertainty* areas in continuous spatio-temporal queries. We define the *uncertainty* area of a query Q as the spatial area that may contain potential moving objects that satisfy Q, with Q not being aware of contents of this area. Uncertainty areas may result in erroneous query results in both moving and stationary contiguous spatio-temporal queries.

- **Moving queries.** Figure 2 gives example *uncertainty* areas that result from moving range queries. Figure 2a represents a snapshot at time T_0 where point P is outside the area of query Q. Thus, P is not physically stored in the database. At time T_1 (Figure 2b), Q is moved to cover a new spatial area. The shaded area in Q represents Qs *uncertainty* area. Although P is inside the new query region, P is not reported in the query answer, simply, because P is not stored in the database. At T_2 (Figure 2c), object P moves out of the query region. Thus, P is never reported at the query result, although it was physically inside the query region in the interval $[T_1, T_2]$.
- **Stationary queries.** Figure 3 gives an example *uncertainty* area in stationary k-nearest-neighbor queries ($k = 2$). At time T_0 (Figure 3a), the query Q has P_1 and P_3 as its answer. P_2 is outside the query spatial region, thus P_2 is not stored in the database. At T_1 (Figure 3b), P_1 moves far from Q. Since Q is aware of P_1 and P_3 only, we extend the spatial region of Q to include the new location of P_1. Thus, an *uncertainty* area is produced. Notice that Q is unaware of P_2 since P_2 is not stored in the database. At T_2 (Figure 3c), P_2 moves out of the new query region. Thus, P_2 never appears as an answer to Q, although P_2 should have been part of the answer in the time interval $[T_1, T_2]$.

2.3 Avoiding Uncertainty by Caching

In PLACE, we avoid *uncertainty* areas in continuous spatio-temporal queries using a *caching* technique. The main idea is to predict the uncertainty area of a continuous query Q and *cache* in-memory all moving objects that lie in Q's uncertainty area. Whenever an uncertainty area is produced, we probe the in-memory *cache* and produce the result immediately. A *conservative* approach for *caching* is to expand the query region in all directions with the maximum possible distance that a moving object can travel between any two consecutive updates. Such *conservative* approach completely avoids uncertainty areas where it is guaranteed that all objects in the uncertainty area are stored in the *cache*.

3 Spatio-temporal Operators

In the PLACE server we encapsulate query processing algorithms inside physical pipelined query operators that can be part of a query execution plan. By having pipelined query operators, we achieve three goals: (1) Spatio-temporal operators can be combined with other operators (e.g., distinct, aggregate, and join operators) to support incremental evaluation for a wide variety of continuous spatio-temporal queries. (2) Pushing spatio-temporal operators deep in the query execution plan reduces the number of tuples in the query pipeline. This reduction comes from the fact that spatio-temporal operators act as filters to the above operators. (3) Flexibility in the query optimizer where multiple candidate execution plans can be produced.

The main idea of spatio-temporal operators is to keep track of the recently reported answer of each query Q in a query buffer termed $Q.Answer$. Then, for each newly incoming tuple P, we perform two tests: Test I - Is P part of the previously reported $Q.Answer$? Test II - Does P qualify to be part of the current answer? Based on the results of the two tests, we distinguish among four cases:

- **Case I:** P is part of $Q.Answer$ and P still qualify to be part of the current answer. As we process only the updates of the previously reported result, P will not be processed.
- **Case II:** P is part of $Q.Answer$, however, P does not qualify to be part of the answer anymore. In this case, we report a *negative* update P^- to the above query operator.
- **Case III:** P is not part of $Q.Answer$, however, P qualifies to be part of the current answer. In this case, we report a *positive* update to the above query operator.
- **Case IV:** P is not part of $Q.Answer$ and P still does not qualify to be part of the current answer. In this case, P has no effect on Q.

Similarly, whenever a query reports movement, it classifies in-memory stored objects into four categories C_1 to C_4 as follows:

- $C_1 \subset Q.Answer$ and C_1 satisfies the new $Q.Region$. Such objects are not processed where they keep their status as part of the query answer.
- $C_2 \subset Q.Answer$, C_2 does not satisfy the query region. For each data object in C_2, we report a *negative* update.
- $C_3 \not\subset Q.Answer$ and C_3 satisfies the new $Q.Region$. For each data object in C_3, we report a *positive* update.
- $C_4 \not\subset Q.Answer$ and C_4 does not satisfy $Q.Region$. Such objects are not processed where they keep their status as they do not belong to the query answer.

3.1 Generic Pipelined Query Operators

PLACE utilizes a unified framework (e.g., see [MA05a]) to deal with continuous range queries as well as continuous k-nearest-neighbor queries. In addition, there is no distinction between stationary and moving queries where both of them are treated similarly. The main idea for dealing with moving queries is to treat data and queries similarly. Thus, data as well as queries have the ability to change their location and size over time. For k-nearest-neighbor queries, a kNN query is represented as a circular range query. The only difference is that the size of the query range may grow or shrink based on the movement of the query and objects of interest. Once the kNN query is registered in the PLACE server, the first incoming k objects are considered as the initial query answer. The radius of the circular region is determined by the distance from the query center to the current kth farthest neighbor. Then, the query execution continues as a regular range query, yet with a variable size. Whenever a newly coming object P lies inside the circular query region, P removes the kth farthest neighbor from the answer set (with a *negative* update) and adds itself to the answer set (with a *positive* update). The query circular region is *shrunk* to reflect the new kth neighbor. Similarly, if an object P, that is one of the k neighbors, updates its location to be outside the circular region, we expand the query circular region to reflect the fact that

P is considered the farthest kth neighbor. Notice that in case of expanding the query region, we do not output any updates.

3.2 SQL Syntax

As the PLACE server extends both PREDATOR [Ses98] and NILE [HMA+04], we extend the SQL language provided by both systems to support spatio-temporal operators. A continuous query is registered at the PLACE server using the SQL:

```
REGISTER QUERY query_name AS
SELECT select_clause
FROM from_clause
WHERE where_clause
INSIDE inside_clause
kNN knn_clause
WINDOW window_clause
```

The `REGISTER QUERY` statement registers the continuous query at the PLACE server with the *query_name* as its identifier. A continuous query is dropped form the system using the SQL `DROP QUERY` *query_name*. The *select_clause*, *from_clause*, and *where_clause* are the same as those in the PREDATOR [Ses98] database management statement. The *window_clause* is the same as that in the NILE [HMA+04] stream query processor to support continuous sliding window queries [HFAE03]. The *inside_clause* may represent stationary/moving rectangular or circular range queries. Moving queries are tied to *focal* objects. As the *focal* object reports movement update to the server, we update the query region. A rectangular range query can have one of the following two forms:

- Static range query (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) represent the top-left and bottom-right corners of the rectangular range query.
- Moving rectangular range query $('M', ID, xdist, ydist)$, where $'M'$ is a flag to indicate that the query is moving, ID is the identifier of the query *focal* point, and $xdist$ and $ydist$ are the length and width of the query rectangle.

A circular range query has the same syntax except that we define only the radius instead of (x, y). Similarly, the *knn_clause* for continuous k-nearest-neighbor queries may have one of the following two forms:

- Static kNN query (k, x, y), where k is the number of neighbors to be maintained, and (x, y) is the center of the query point.
- Moving kNN query $('M', k, ID)$, where $'M'$ is a flag to indicate that the query is moving, k is the number of neighbors to be maintained, and ID is the identifier of the query *focal* point.

4 Shared Execution

In a typical spatio-temporal application (e.g., location-aware services), there are large numbers of concurrent spatio-temporal continuous queries. Dealing with each query as a separate entity would easily consume the system resources and degrade the system performance. Figure 4a gives the pipelined execution of N queries (Q_1 to Q_N) of various types with no sharing, i.e., each query is considered a separate entity. The input data stream goes through each spatio-temporal query operator separately. With each operator, we keep track of a separate buffer that contains all the objects that are needed by this query (e.g., objects that are inside the query

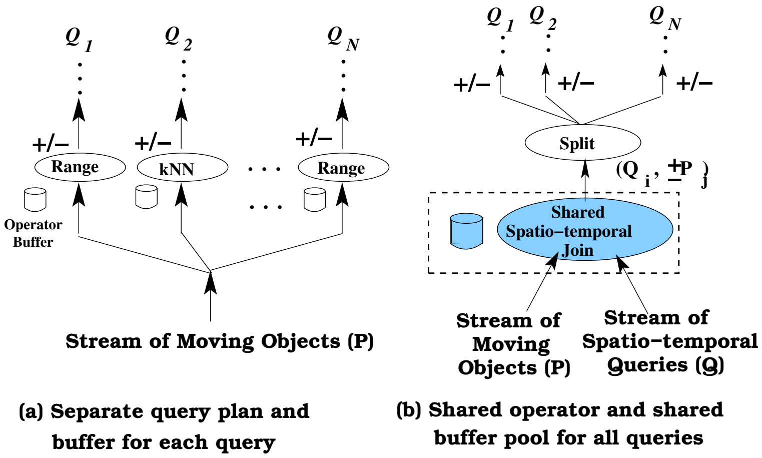


Figure 4: Overview of shared execution in the PLACE server.

region or its cache area). With a separate buffer for each single query, the memory can be exhausted with a small number of continuous queries. Figure 4b gives the pipelined execution of the same N queries as in Figure 4a, yet with the shared pipelined query operator [MA05b]. The problem of evaluating concurrent continuous queries is reduced to a spatio-temporal join between two streams; a stream of moving objects and a stream of continuous spatio-temporal queries. The *shared* spatio-temporal join operator has a shared buffer pool that is accessible by all continuous queries. The output of the *shared* query operator has the form $(Q_i, \pm P_j)$ that indicates the addition or removal of object P_j to/from query Q_i. The shared query operator is followed by a *split* operator that distributes the output either to the users or to the various query operators.

5 Load Shedding

Even with the shared execution paradigm in PLACE, the memory resource may be exhausted at intervals of unexpected massive numbers of queries and moving objects (e.g., during rush hours). To cope with such intervals, the PLACE server is equipped with a *self-tuning* approach that tunes the memory load to support a large number of concurrent queries, yet with an approximate answer. The main idea is to tune the definition of *significant* objects based on the current workload. By adapting the definition of *significant* objects, the memory load will be *shed* in two ways: (1) In-memory stored objects will be revisited for the new meaning of *significant* objects. If an *insignificant* object is found, it will be *shed* from memory. (2) Some of the newly input data will be *shed* at the input level.

Figure 5 gives the architecture of *self-tuning* in the PLACE server. Once the shared join operator incurs high resource consumption, e.g., the memory becomes almost full, the join operator triggers the execution of the *load shedding* procedure. The *load shedding* procedure may consult some statistics that are collected during the course of execution to decide on a new meaning of *significant* objects. While the shared join operator is running with the new definition of *significant* objects, it may send updates of the current memory load to the *load shedding* procedure. The load shedding procedure replies back by continuously adopting the notion of *significant* objects based on the continuously changing memory load. Finally, once the memory load returns to a stable state, the shared join operator retains the original meaning of *significant* objects and stops the execution of the *load shedding* procedure. Solid lines in Figure 5 indicate the mandatory steps that should be taken by any *load shedding* technique. Dashed lines indicate a set of operations that may or may not be employed based on the underlying *load shedding* technique.

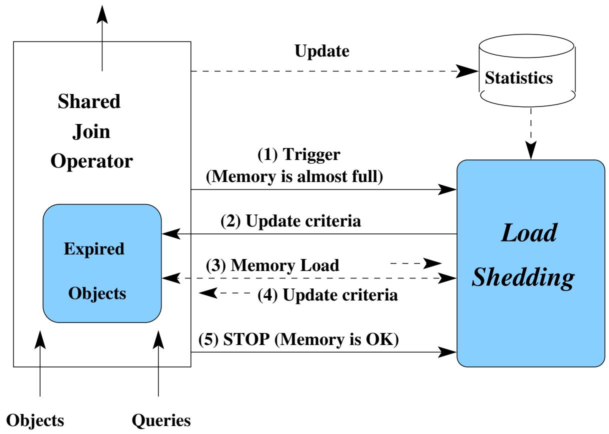


Figure 5: *Self-tuning* in the PLACE server

6 Conclusion

In this paper, we presented the PLACE (Pervasive Location-Aware Computing Environments) server; a database server for location-aware environments developed at Purdue University. The PLACE server combines the recent advances in spatio-temporal query processors and data stream management systems to provide a location-aware database server that efficiently execute large number of concurrent continuous spatio-temporal queries over spatio-temporal data streams. The PLACE server is realized by extending both the PREDATOR database management system and the NILE stream query processor to support the following main features: (1) New physical *spatio-temporal query operators* that can interact with traditional query operators in a large query plan. (2) *Incremental evaluation* through the concepts of positive and negative updates. (3) *Shared execution* of large number of concurrent continuous spatio-temporal queries. (4) *Load shedding* techniques to cope with time intervals of high workload of incoming data objects and/or queries. (5) Unified framework for a wide variety of continuous spatio-temporal queries.

Acknowledgments

This work was supported in part by the National Science Foundation under Grants IIS-0093116, IIS-0209120, and 0010044-CCR. The authors also would like to thank Hicham Elmongui, Thanaa Ghanem, Susanne Hambrusch, Moustafa Hammad, Sunil Prabhakar, and Xiaopeng Xiong for contributing in different stages in realizing the PLACE server.

References

[AHP03] Walid G. Aref, Susanne E. Hambrusch, and Sunil Prabhakar. Pervasive Location Aware Computing Environments (PLACE). http://www.cs.purdue.edu/place/, 2003.

[CEP03] V. Prasad Chakka, Adam Everspaugh, and Jignesh M. Patel. Indexing Large Trajectory Data Sets with SETI. In *Proc. of the Conf. on Innovative Data Systems Research, CIDR*, 2003.

[CM03] Graham Cormode and S. Muthukrishnan. Radial Histograms for Spatial Streams. Technical Report DIMACS TR: 2003-11, Rutgers University, 2003.

[EMA05] Hicham G. Elmongui, Mohamed F. Mokbel, and Walid G. Aref. Spatio-Temporal Histograms. In *SSTD*, 2005.

[GAE05] Thanaa M. Ghanem, Walid G. Aref, and Ahmed K. Elmagarmid. Exploiting Predicate-window Semantics over Data Streams. *SIGMOD Record. To Appear*, 2005.

[GO03] Lukasz Golab and M. Tamer Ozsu. Issues in data stream management. *SIGMOD Record*, 32(2):5–14, 2003.

[HFAE03] Moustafa A. Hammad, Michael J. Franklin, Walid G. Aref, and Ahmed K. Elmagarmid. Scheduling for shared window joins over data streams. In *VLDB*, 2003.

[HMA+04] Moustafa A. Hammad, Mohamed F. Mokbel, Mohamed H. Ali, Walid G. Aref, Ann C. Catlin, Ahmed K. Elmagarmid, Mohamed Eltabakh, Mohamed G. Elfeky, Thanaa M. Ghanem, Robert Gwadera, Ihab F. Ilyas, Mirette Marzouk, and Xiaopeng Xiong. Nile: A Query Processing Engine for Data Streams (Demo). In *ICDE*, 2004.

[HS04] John Hershberger and Subhash Suri. Adaptive Sampling for Geometric Problems over Data Streams. In *PODS*, 2004.

[MA05a] Mohamed F. Mokbel and Walid G. Aref. GPAC: Generic and Progressive Processing of Mobile Queries over Mobile Data. In *MDM*, 2005.

[MA05b] Mohamed F. Mokbel and Walid G. Aref. SOLE: Scalable Online Execution of Continuous Queries on Spatio-temporal Data Streams. Technical Report TR CSD-05-016, Purdue University Department of Computer Sciences, July 2005.

[MGA03] Mohamed F. Mokbel, Thanaa M. Ghanem, and Walid G. Aref. Spatio-temporal Access Methods. *IEEE Data Engineering Bulletin*, 26(2), 2003.

[MXA04a] Mohamed F. Mokbel, Xiaopeng Xiong, and Walid G. Aref. SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases. In *SIGMOD*, 2004.

[MXA+04b] Mohamed F. Mokbel, Xiaopeng Xiong, Walid G. Aref, Susanne Hambrusch, Sunil Prabhakar, and Moustafa Hammad. PLACE: A Query Processor for Handling Real-time Spatio-temporal Data Streams (Demo). In *VLDB*, 2004.

[MXHA04] Mohamed F. Mokbel, Xiaopeng Xiong, Moustafa A. Hammad, and Walid G. Aref. Continuous Query Processing of Spatio-temporal Data Streams in PLACE. In *Proceedings of the second workshop on Spatio-Temporal Database Management, STDBM*, 2004.

[PCC04] Jignesh M. Patel, Yun Chen, and V. Prasad Chakka. STRIPES: An Efficient Index for Predicted Trajectories. In *SIGMOD*, 2004.

[Ses98] P. Seshadri. Predator: A Resource for Database Research. *SIGMOD Record*, 27(1):16–20, 1998.

[TPZL05] Yufei Tao, Dimitris Papadias, Jian Zhai, and Qing Li. Venn Sampling: A Novel Prediction Technique for Moving Objects. In *ICDE*, 2005.

MOBI-DIC: MOBIle DIscovery of loCal Resources in Peer-to-Peer Wireless Network *

Hu Cao, Ouri Wolfson, Bo Xu and Huabei Yin

Department of Computer Science
University of Illinois at Chicago
{*hcao2, wolfson, boxu, hyin*}@*cs.uic.edu*

Abstract

In this paper we examine management of databases distributed among moving objects. The objects are interconnected by a Mobile Ad Hoc Network. Several inherent characteristics of this environment, including the dynamic and unpredictable network topology, the limited peer-to-peer communication bandwidth, and the need for incentive for peer-to-peer cooperation, impose challenges to data management. In this paper we discuss these challenges in the context of a database that represents resource information. The information is disseminated and queried by the moving objects in search of resources. We are currently building such a resource discovery engine called MOBI-DIC: MOBIle DIscovery of LoCal Resources.

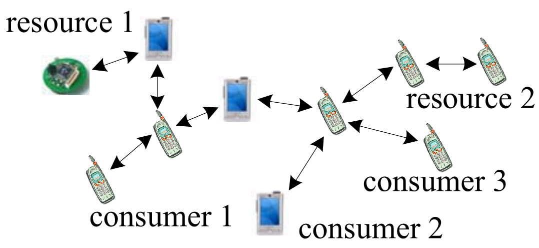
MOBI-DIC will enable quick building of matchmaking or resource discovery services in many application domains, including social networks, transportation, mobile electronic commerce, emergency response, and homeland security. For example, in a large professional, political, or social gathering, the technology is useful to automatically facilitate a face-to-face meeting based on matching profiles. In transportation, MOBI-DIC incorporated in navigational devices can be used to disseminate to other similarly-equipped vehicles information about relevant resources such as free parking slots, traffic jams and slowdowns, available taxicabs, and ride sharing. In mobile electronic commerce, MOBI-DIC is useful to match buyers and sellers in a mall, or to disseminate information about a marketed product. In emergency response, MOBI-DIC can be used by first responders to support rescue efforts (locate victims, and match responder capability with needs) even when the fixed infrastructure is inoperative. In homeland security, sensors mounted on neighboring containers can communicate and transitively relay alerts to remote check-points.

1 Introduction

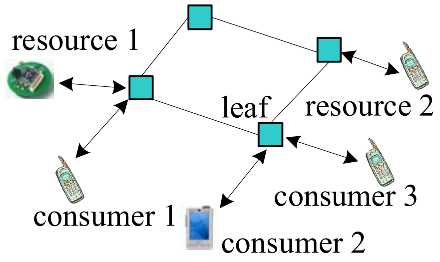
Mobile local search is a procedure in which a mobile user searches for local resources, i.e. resources that are in geographic proximity to the user. In many situations, the local resources that are of interest to mobile users

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*This research is supported by NSF Grants 0326284, 0330342, ITR- 0086144, 0513736, and 0209190.



(a) Mobile ad hoc solution to local search. The nodes are sensors, PDA's, laptops, cell phones.



(b) Infrastructure based solution to mobile local search.

Figure 1: Two mobile local search solutions.

are only available during a limited period of time and these resources themselves may be mobile. For example, a cab driver wants to find a customer near by. The customer may be moving and she is available only until she hires a cab. Similarly, the current traffic speed on a road segment, the available parking slots around a driver, the available workstations in a large convention hall, are temporarily valid or available resources. We call these *spatio-temporal resources*.

Mobile local search for spatio-temporal resource is a special case of resource discovery (see e.g. [6, 9]) and publish/subscribe [14] applications, in the sense that the resources or events are spatio-temporal and limited in geographic scope. However, these characteristics make them less amenable to a centralized or hierarchical solution using the infrastructure. The reason is that installing a server in many local areas, and accessing these servers through the cellular infrastructure may be expensive. Furthermore, the temporary nature of the resources makes update and query response time critical, which again necessitates a large number of servers.

Therefore, in our project we explore a peer-to-peer (P2P) approach to local search that does not require a central server or a wireless infrastructure. However, when the infrastructure is available, the proposed P2P paradigm can augment it to make the search more efficient.

In our proposed approach (see Figure 1), a set of moving objects (denoting processors and sensors, some of which may be static) form a mobile ad-hoc network. Some moving objects are consumers and some are resources, and they communicate with each other via short-range wireless technologies such as IEEE 802.11, Bluetooth, or Ultra Wide Band (UWB). With such communication mechanisms, a moving object receives resource information from its neighbors, or from remote objects by multi-hop transmission relayed by intermediate moving objects. The cellular and mobile ad hoc approaches can be combined into an architecture in which resource-information disseminated in a mobile ad-hoc network augments the infrastructure by covering the areas that are not be covered by the hotspots, and it enhances the local search capability offered by the hotspots in areas that are covered by the infrastructure. In other words, the P2P approach can also be used to communicate among the leaves of a hierarchical cellular architecture (see Figure 1(a), further enhancing the search capability. In Figure 1(b), rectangles are leaves of a possibly fixed hierarchical infrastructure, each of which controls an area called a ”cell”.

We are currently building a software platform that supports mobile search and discovery of spatio-temporal resources in mobile ad-hoc networks, possibly in conjunction with the existing infrastructure. We call this platform MOBI-DIC (MOBIle DIscovery of loCal resources). MOBI-DIC is based on a local communication paradigm in a peer-to-peer network. Conceptually, the paradigm consists of neighbors exchanging reports, where each report is a resource description or a query. When a moving object m receives new reports, it incorporates them into its local reports database, and broadcasts the new reports database to its neighbors. Upon receiving the broadcast from m, each neighbor incorporates the received reports into its own local reports database, and broadcasts the new reports database. Thus reports transitively spread across network nodes. With this local broadcasting, the size of m's local database may continuously increase as new reports are generated. In order to limit the broadcast volume, we employ relevance functions that prioritize reports. Each moving object broadcasts only the k most relevant (top k) reports during a local broadcast. In other words, ranking is used to control flooding. We call this *paradigm rank-based broadcast (RBB)*.

It is important to note that rank-based broadcast reduces communication of unimportant and spam information. Furthermore, the user is shielded from unimportant, unwanted, and spam information by his/her query; in other words, the user is notified by his/her processor only of information that satisfies the user's query.

Compared to existing resource discovery protocols which rely heavily on constructed routing structures (see e.g. [6, 9, 14]), rank-based broadcast adapts better to high mobility and variance in network connectivity. Indeed, in a highly dynamic network that is susceptible to partitions, the constructed routing structure may easily become obsolete; whereas rank based broadcast does not rely on global structures, and a node does not need to know the global network topology. Furthermore, with the appropriate relevance function, rank-based broadcast provides spatial and temporal proximity awareness. In other words, the relevance of a report, and consequently its transmission probability, decays as it becomes older and it travels away from its point of origin.

MOBI-DIC will enable quick building of matchmaking or resource discovery services in many application domains including social networks, transportation, mobile electronic commerce, emergency response, and homeland security. For example, in a large professional, political, or social gathering, the technology is useful to automatically facilitate a face-to-face meeting based on matching profiles. In transportation, MOBI-DIC incorporated in navigational devices can be used to disseminate to other similarly-equipped vehicles information about relevant resources such as free parking slots, traffic jams and slowdowns, available taxicabs, and ride sharing. In mobile electronic commerce, MOBI-DIC is useful to match buyers and sellers in a mall, or to disseminate information about a marketed product. In emergency response, MOBI-DIC can be used by first responders to support rescue efforts even when the fixed infrastructure is inoperative; it will match specific needs with expertise (e.g. burn victim and dermatologist), and help locate victims. In homeland security, sensors mounted on neighbouring containers can communicate and transitively relay alerts to remote check-points.

In summary, MOBI-DIC provides mobile users a search engine for transient and highly dynamic information in a local geospatial environment. MOBI-DIC employs a unified model for both the cellular infrastructure and the mobile ad hoc environments. When the infrastructure is available, it can be augmented by the mobile ad hoc approach.

2 Concepts and Model

In MOBI-DIC there are *moving objects* and *resources* (also called events in publish-subscribe terminology). Each moving object m may generate resource descriptions for resources it learns or produces. A resource description for a resource R is denoted by $a(R)$. Each resource description $a(R)$ has a set of attributes. In addition to application specific attributes, all the resource descriptions for spatio-temporal resources have two common attributes, namely *resource-time*, and *resource-location*. Resource-time is the time when the resource description is generated. Resource-location is the location of R. If R is a mobile resource, then the location is a (possibly uncertain) trajectory that gives the current location as a function of time.

Each moving object m may also generate queries (or subscriptions in publish-subscribe terminology) which express m's interests in certain types of resources. A query may be associated with a function that computes the matching degree between the query and a resource description. We term a query and a resource description collectively as a *report*.

Each moving object m has a database that stores the reports that it has generated or has received from other moving objects. This is called the *reports database*. m is called the *producer* of the reports that it generates, the *consumer* of the reports that it is interested in receiving, and the *broker* of the reports it neither generates nor is interested in receiving. Each report has a *relevance* at location l and at time t that is determined by a *relevance*

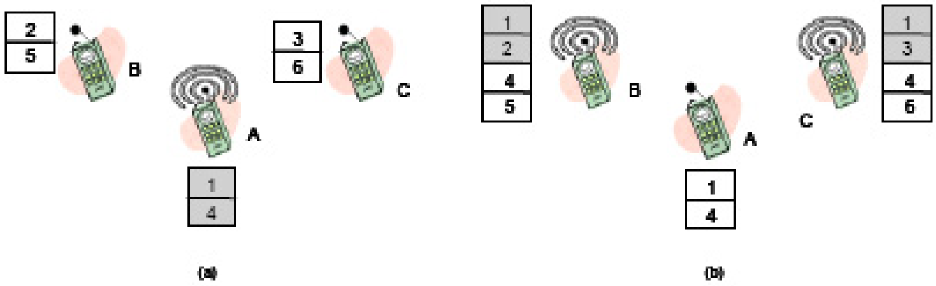


Figure 2: (a) Moving object *A* broadcasts its top two reports which are reports 1 and 4. (b) After receiving from *A*, object *B* incorporates the received reports, re-ranks, and broadcasts the top two (shadowed). The same for *C*.

function. The relevance of a report determines the priority of the report relative to other reports in the database, in allocating the available bandwidth of a particular broadcast. The relevances of all the reports are calculated for a specific location point and a specific time point, when it is determined that a broadcast has to be performed at those coordinates.

Observe that communication in mobile ad hoc networks is unreliable and disconnection may occur at any time, particularly in highly mobile environments such as vehicular networks. Furthermore, due to collisions in a dense environment, and due to energy considerations, the bandwidth available to each moving object may be limited. Therefore we postulate that ranking of reports is important, so that the most important reports are transmitted first. Therefore we propose a paradigm, *rank-based broadcast* (RBB), in which each moving object m periodically sorts the reports in its local database according to the relevance function, and broadcasts top k to its neighbors (i.e. the moving objects within the transmission range from m). The frequency of broadcasting and the size of each broadcast (i.e. k) depend on the available bandwidth and power. Figure 2 gives an example of the rank-based broadcast procedure.

Before ending this section we comment on the relationship between the RBB and flooding protocols. When energy and bandwidth are abundant, then RBB naturally reduces to flooding in the sense that each moving object caches all the reports that it receives, and relays them to each other moving object that it encounters. However, observe that as new resources are generated, it becomes increasingly unlikely that a moving object has enough power and bandwidth to relay all the reports it has ever received. Then a prioritization mechanism, such as that provided by RBB, becomes critical.

3 Results from Preliminary Work

In our prior work [29, 28, 26, 25, 27, 30, 31] we analyzed the way a report is propagated in geospace and time, the benefit of information dissemination in capturing competitive resources (i.e. resources that can be used by only one user at a time, such as a parking slot), how well the average local database reflects the status of physical resources, and how our approach compares with the central server model and with existing work on publish/subscribe in wireless ad-hoc networks.

3.1 Pattern of Report Propagation

In [29] we studied a simple situation where there is only one resource type in the system and the relevance is determined solely by a spatio-temporal function. With this function, the relevance of a report decreases as the time elapsed from its creation (i.e. age) increases, and as the distance from the resource increases. Our results show that when the memory of each moving object is limited and there are enough resources in the system, then the propagation of a report diminishes as the age and the distance increase; flooding is thus automatically limited to spatial proximity to the resource, and temporal proximity to the initial transmission-time of the report by the resource. Furthermore, the spatial and temporal boundaries automatically adapt depending on the number of resources in the system, the traffic density and speed, and other parameters that dictate the amount of storage, processing power, and bandwidth that should be allocated to each resource. For example, if the number of resources is small, each report will stay in the system longer, and spread farther.

3.2 Benefit of Resource Information

We analyzed the value of resource information in terms of how much time is saved when using the information to acquire competitive resources compared to when not using the information (see [28]). The results show that using information always reduces acquisition time compared to blind search (i.e. the information is not used). The value of information varies depending on the number of resources in the system, the traffic density and speed, and the wireless transmission range etc. In some cases by using the resource reports acquisition time is cut by more than 75%. We also developed information usage strategies to alleviate the herding effect, which occurs when a number of users all receive the same report and all decide to go to the same resource.

3.3 Accuracy of Information

We define the accuracy as the percentage of time when the report at a certain rank position is valid (i.e. the report correctly reflects the actual availability status of the reported resource). We evaluated by simulations the accuracy of information under different system environments (see [26]). The results show that in most situations, the accuracy of the first rank position is more than 85%.

We also compared by simulations the mobile ad-hoc approach and the central server approach in terms of accuracy. We found that with very reasonable object density and wireless transmission range, the accuracy of mobile ad-hoc approach reaches that of the central server approach. This indicates that the mobile ad-hoc approach could serve well as an alternative to the central server approach but with much less operational cost.

3.4 Comparison with Publish/Subscribe in Wireless Ad-hoc Networks

We compared RBB with Publish/Subscribe Tree (PSTree), a publish/subscribe algorithm designed for wireless ad-hoc networks (see [14]). We compared the two algorithms in terms of two aspects: (1) communication efficiency, namely the total relevance of resource descriptions that is delivered to consumers for each unit of bandwidth consumed; and (2) the total relevance of the resource descriptions delivered to consumers, regardless of communication cost,. The results show that in most cases RBB outperforms PSTree, in both communication efficiency and total relevance. We found that in a high mobility environment, since the network topology changes frequently, PSTree has to consume a lot of communication to update the routing structure. Furthermore, if the topology changes frequently, then relevant reports are lost anyway. However, RBB does not rely on global structures in the network, only on statistical information which is used to determine the relevance of a report.

4 Summary of the On-going Research

We are studying the following problems in this project:

1. Performance Measures. We are designing performance measures suitable for local search in a mobile ah-hoc network.

2. Report Ranking. We are investigating techniques that rank the reports in a local database. These techniques provide a total rank in terms of relevance-to-neighbors for the reports across all the resource types,

including queries, stored in a moving object's reports database. The key issue is how to quantify the relative contributions of different attributes to the utility of a report.

3. Rank-based Broadcast. We are studying variants of the rank-based broadcast conceptual model. These variants differ in their strategies of when to broadcast and what to broadcast.

4. Query Answering. We are studying how to deliver the match discovered at a broker to the query originator.

5. Power Management. We are incorporating power conservation mechanisms to RBB for mobile devices such as PDA and sensors.

6. Heterogeneous Environments. We are studying the RBB paradigm in environments where devices have different short range wireless capabilities, such as 802.11 and Bluetooth.

7. Augmenting the infrastructure. We are investigating methods for integrating mobile ad-hoc networks with cellular network.

8. Incentive Mechanisms. We are studying incentive mechanisms that stimulate moving objects to cooperate in report dissemination.

9. Security. We are studying solutions to various attacks from malicious or selfish parties. The solutions should minimally compromise privacy, and make minimum assumptions concerning tamper-resistant components.

10. Report Language. We are developing a language for the expression of resource description and queries.

11. Evaluation Tools. In order to evaluate our approach, we are developing three types of tools, including analytical models, a simulation testbed, and a proof-of-concept prototype.

5 Relevant Work

Resource Discovery and Publish/Subscribe Literature. The existing methods of resource discovery can be categorized into two types. Methods of one type are those relying on dedicated directory agents, e.g. SLP, Jini, Salutation, and UPnP. However, it is not clear how to make these agents available and accessible in a mobile ad hoc environment. Another type of resource discovery methods uses the peer-to-peer paradigm, however these solutions are based on fixed peers embedded in the infrastructure.

Resource discovery and publish/subscribe in mobile ad hoc networks are usually implemented by building a routing structure for resource information (see e.g. [6, 9, 14]). Most of this work uses the "pull" mode, where a moving object searches the network for resources. As we have shown in the preliminary work, pull can be inefficient, particularly when the network topology is highly dynamic.

Work has also been done on data dissemination in mobile peer-to-peer networks [20, 21]. These methods use the gossiping/epidemic communication paradigm. However, they consider dissemination of regular data objects rather than spatio-temporal resources, and they do not rank the resources for determining what to broadcast.

Social Serendipity ([7]) is another related project. However, it focuses exclusively on social networks, whereas MOBI-DIC proposes a more general local search platform. Furthermore, the data dissemination and matchmaking in MOBI-DIC are totally decentralized, whereas Social Serendipity uses a central server approach.

Data Broadcasting. In data broadcasting [1, 2], data is periodically pushed from a server to a group of clients over a broadcast or multicast link. The major issue is how to schedule broadcast content to minimize the access time and tuning time of the client community. For example, the data disks [1, 2] scheme broadcasts hot (popular) data items with higher frequency than cool (unpopular) data items. We use the same idea in RBB. However, since we have a mobile ad-hoc environment and there are no dedicated servers, we enable the awareness of popularity by disseminating queries; In other words, we broadcast both data and queries. We divide the bandwidth capacity between resource descriptions and queries. Furthermore, in our model the importance of a resource description depends not only on its popularity, but also on many other factors such as its age and the resource location.

Mobile Ad-hoc Networks. The work in this area mainly concerns with sending a message to a specific destination given by the network address ((see [22] for a survey). In our case the network addresses of the destinations (i.e. consumers) are not known a priori. There is a body of work that deals with geographic routing (e.g. [15]) in which a message is routed from a source node to a geographic location or area. However, in most of the existing works in this area, message delivery is possible only if the source and destination are connected, namely there exists a path from the source to the destination. Li [18], Chen [5], and Vahdat [23] are among the first few works on routing in disconnected ad hoc networks. They either do not fit our context (e.g. [18] requires a moving object to actively move to reach the next object), or are too aggressive on bandwidth consumption (e.g. [23] uses flooding).

Mobile-to-mobile and Mobile-to-roadside Communication. CarTALK 2000 [4], VICS [24], and FleetNet [8] are projects aimed at designing, testing and evaluating co-operative driver assistance systems based upon vehicle-to-vehicle communication in order to improve the overall safety and convenience of the traffic participants. Infostations [10] is a mobile-to-roadside communication system that smartly allocates wireless channel depending on the positions of moving objects relative to infostations (i.e. hotspots). Our project can be considered an application that provides data management capabilities (data collection, organization, integration, modeling, dissemination, querying) on top of the underlying communication system provided by these systems.

Incentive Mechanisms. Incentive mechanisms have been studied in mobile ad-hoc networks (see e.g. [3, 33]) for stimulating intermediate nodes to forward messages to a given destination (rather than resource dissemination). We expect to be able to draw on existing ideas in this area. Incentive mechanisms have also been studied for static peer-to-peer networks (see e.g. [17]). In this case the static nature of the problem is often relied upon heavily, for example, by 'punishing' a user that is found noncooperative over time. Such a longer-time perspective is missing in our mobile ad-hoc environment, which may rarely involve the same pair of moving objects in an exchange.

Static Sensor Networks. A database approach has been applied to static sensor networks in Cougar [32], TinyDB [12], and direct diffusion [16]. Our distributed query processing relies on opportunistic interactions between mobile nodes, and is totally different.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast Disks: Data Management for Asymmetric Communication Envrionments. *Proc. ACM SIGMOD Conf.*, San Jose, CA, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik. Balancing Push and Pull for Data Broadcast. *Proc. ACM SIGMOD Conf.*, Tucson, AZ, May, 1997.

[3] L. Buttyan and J. Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology - Lausanne, Department of Communication Systems, 2001.

[4] CarTalk 2000, website: http://www.cartalk2000.net/

[5] X. Chen and A. L. Murphy. Enabling Disconnected Transitive Communication in Mobile Ad Hoc Networks, *ACM Principals of Mobile Computing, POMC'01*, 2001.

[6] S.M. Das, H. Pucha, and Y. Hu. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. In *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*, English Lake District, UK, Dec. 2004.

[7] N. Eagle and A. Pentland. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing*, Special Issue: The Smart Phone. April-June 2005. pp 28-34.

[8] FleetNet-Internet on the Road project, website: http://www.fleetnet.de

[9] C. Frank and H. Karl. Consistency Challenges of Service Discovery in Mobile Ad Hoc Networks. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 105-114, Venice, Italy, 2004.

[10] R.H. Frenkiel, B.R. Badrinath, J. Borras, and R. Yates. The Infostations Challenge: Balancing Cost and Ubiquity in Delivering Wireless Data. *IEEE Personal Communications Magazine*, 7(2): pp. 66-71, April 2000.

[11] I. Gruber, R. Schollmeier, and W. Kellerer. Performance Evaluation of the Mobile Peer-to-Peer Protocol. In *Proceedings of Fourth International Workshop on Global and Peer-to-Peer Computing*, Chicago, USA, Apr. 2004.

[12] J. Hellerstein, W. Hong, S. Madden and K. Stanek. Beyond Average: Toward Sophisticated Sensing with Queries. *The Second International Workshop on Information Processing in Sensor Networks*, 2003.

[13] Y. C. Hu and S. M. Das. Exploiting the Synergy between Peer-to-Peer and Mobile Ad Hoc Networks. In *HotOS-IX*, 2003.

[14] Y. Huang and H. Garcia-Molina. Publish/subscribe in a Mobile Environment. In *Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE01)*, Santa Barbara, CA, May 2001.

[15] T. Imielinski, J. Navas. GPS-Based Geographic Addressing, Routing, and Resource Discovery. *Communications of the. ACM*, Vol. 42, No. 4, pp 86-92, 1999.

[16] C. Intanagonwiwat, R. Govindan and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of MobiCOM*, Boston, MA, 2000.

[17] K. Lai, M. Feldman, I. Stoica and J. Chuang. Incentives for Cooperation in Peer-to-Peer Networks. *1st Workshop on the Economics of P2P Systems*, 2003.

[18] Q. Li and D. Rus. Sending Messages to Mobile Users in Disconnected Ad-hoc Wireless Networks. *MOBICOM*, 2000.

[19] M. Mamei and F. Zambonelli. Location-based and Content-based Information Access in Mobile Peer-to-Peer Computing: the TOTA Approach. In *Second International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2003)*, Melbourne, Australia, 2003.

[20] M. Papadopouli and H. Schulzrinne. Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination among Mobile Devices. In *MobiHoc*, 2001.

[21] F. Perich, A. Joshi, T. Finin, and Y. Yesha. On Data Management in Pervasive Computing Environments. *IEEE Transactions on Knowledge and Data Engineering*, 16(5), May 2004, pp. 621-634.

[22] E. Royer and C. Toh, A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks, *IEEE Personal Communications*, pages 46-55, Apr. 1999.

[23] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks, Technical Report CS-200006, Duke University, April 2000.

[24] Vehicle Information and Communication System Center (VICS Center), website: http://www.vics.or.jp/eng

[25] O. Wolfson, A. Ouksel, and B. Xu. Resource Discovery in Disconnected Mobile Ad-Hoc Networks. In *International Workshop on Next Generation Geospatial Information*, Cambridge, MA, 2003.

[26] O. Wolfson, B. Xu. Opportunistic Dissemination of Spatio-temporal Resource Information in Mobile Peer-to-Peer Networks. In *1st International Workshop on P2P Data Management, Security and Trust (PDMST'04)*, 2004.

[27] O. Wolfson and B. Xu. Data-on-the-road in Intelligent Transportation Systems. In *IEEE International Conference on Networking, Sensing, and Control (ICNSC 2004)*, Taipei, Taiwan, 2004.

[28] O. Wolfson, B. Xu, and H. Yin. Dissemination of spatial-temporal information in mobile networks with hotspots. In *Proceedings of the Second International Workshop On Databases Information Systems and Peer-to-Peer Computing*, 2004.

[29] B. Xu, A. Ouksel, O. Wolfson, Opportunistic Resource Exchange in Inter-vehicle Ad Hoc Networks. In *Proc. of 2004 IEEE International Conference on Mobile Data Management*, Berkeley, California, Jan. 2004.

[30] B. Xu and O. Wolfson. Privacy Solutions by Mobile Peer-to-Peer Communication. In *Center for Intellectual Property Law and Info. Tech., (CIPLIT) Symposium*, Chicago, Illinois, 2004.

[31] B. Xu, O. Wolfson. Data Management in Mobile Peer-to-Peer Networks. Springer Verlag Lecture Notes in Computer Science. In *Proc. of the 2nd International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P'04)*, Toronto, Canada, Aug. 2004.

[32] Y. Yao and J. Gehrke. Query Processing in Sensor Networks. *First Biennial Conference on Innovative Data Systems Research*, 2003.

[33] S. Zhong, J. Chen and Y. Yang. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. *In Proceedings of IEEE INFOCOM* 2003.

Energy-Efficient Monitoring of Spatial Predicates over Moving Objects

Haibo Hu   Dik Lun Lee
Department of Computer Science
Hong Kong University of Science and Technology
{haibo,dlee}@cs.ust.hk

Abstract

We study the problem of monitoring the changes of predicates on the spatial relations between objects, which is fundamental for many monitoring and tracking applications. As the location update problem is the core of object monitoring, we develop a cost model on the power consumption and propose a family of energy-efficient location update schemes. In these schemes, upon each location update the server sends back additional information about the predicates so that the client can defer the next update until it is about to change the predicate value. As a result, the updates are less frequent than the traditional periodic or dead-reckoning schemes without undermining the monitoring accuracy.

1 Introduction

With the flourishing of location and tracking sensors, monitoring the positions of moving objects becomes a reality. This paves the way for the mass deployment of all kinds of location-aware applications. However, the problem of when or how often the location sensors should report the locations of the objects they are monitoring or tracking has been ignored. These reporting activities are known as "location updates". A large number of location updates not only incur huge network traffic, but pose heavy burden on the servers that manage the location data. In addition, frequent location updates exhaust the sensor power supply quickly, which is a serious issue in mobile and sensor networks. On the other hand, an insufficient number of location updates lead to severe monitoring error.

In the literature, the predominant location update paradigm is the "periodic" scheme, which schedules the updates at regular intervals. However, this scheme cannot scale to large populations and the optimal interval values must be tuned for different systems. It is also inefficient in terms of its monitoring accuracy versus the number of location updates it incurs. In this paper, we first propose the problem of monitoring spatial predicates. These spatial predicates, aligned with those defined in the Open Geospatial Consortium reference model [3], are defined as Boolean assertions on the topological, metric, or directional relations between spatial objects. We then develop the cost model of power consumption for this monitoring problem and propose a family of location update schemes that reduces this cost. As the dynamics of these spatial predicates are interesting to many location-aware monitoring services (e.g. fleet tracking, combat troop tactics) and are also regarded

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

as the primitives of spatial queries, these schemes serve as a general solution to a wide range of monitoring applications. These schemes take the same approach that, along with each location update the server sends back additional information (more specifically the *safe regions*) about the spatial predicates being monitored so that the client can delay the next update until it is about to change the predicate value. In other words, these schemes achieve fewer number of updates without undermining the monitoring accuracy.

The rest of the paper is organized as follows. Section 2 formally defines the spatial predicates and relates them to spatial queries. Section 3 proposes the notion of safe regions and the operations on them. Following the system model in Section 4, we develops the cost model of power consumption in Section 5 based on which the family of update schemes are presented.

2 Spatial Predicates

Definition 1: A **spatial predicate** is a Boolean assertion on the topological, metric, or directional relations between two spatial entities a and b. It is denoted as $a \circ b$, where $\circ$ stands for one of the relations in the following categories:
1. topological relations: "contains" ($\odot$), "overlap" ($\otimes$), "disjoint" ($\oslash$), "within" ($\ominus$), etc;
2. metric relations: "nearest to" ($\Rsh$), "farthest to" ($\dashrightarrow$), "neighboring" ($\curvearrowright$), etc;
3. directional relations: "north of" ($\trianglelefteq$), "south of" ($\trianglerighteq$).
Furthermore, a is called the *subject* and b is called the *observer* of the predicate.

The subject corresponds to a moving object, whereas the observer could be either moving or stationary. By this definition, all conventional spatial queries can be expressed as a combination of these spatial predicates. For example, a nearest neighbor query corresponds to some "nearest to" predicates in which the query point is the observer and a neighborhood object is the subject; a window query corresponds to some "within" and "overlap" predicates in which the query window is the observer; and a spatial join between two datasets corresponds to the predicates in which each object in the second dataset serves as the observer. The formal correspondence of these queries to the spatial predicates on the moving object dataset $\mathcal{S}$ are listed below.

Example 1: 1) Range query with range q: $Range(q) = \{s \in \mathcal{S} | s \curvearrowright q = true\, AND\, (s \ominus q \vee s \otimes q) = true\}$
2) Nearest neighbor query with query point (or query object) q: $NN(q) = \{s \in \mathcal{S} | s \curvearrowright q = true\, AND\, s \Rsh q = true\}$
3) Distance spatial join for $\mathcal{S}$ and another dataset $\mathcal{R}$ within distance threshold δ: $Dist\_Join(\mathcal{S}, \mathcal{R}, \delta) = \{(s, r) | (s \ominus (r \bigoplus \delta) \vee s \otimes (r \boxplus \delta)) = true, r \in \mathcal{R}, s \in \mathcal{S}\}$, where $\boxplus$ stands for the Minkowski sum [4].

As such, continuously monitoring these spatial queries is equivalent to monitoring the changes of the corresponding spatial predicates. Besides spatial queries, many location-aware applications require the notification of certain spatial events, which can be expressed as spatial predicates. The following is a typical example.

Example 2: In a mobile ad-hoc network, mobile client s is the parent of client a, b and c in the routing tree (see Figure 1). In the physical transmission layer, s must reach a, b and c. As such, in case any child moves beyond the effective transmission range δ of s, the routing topology of a, b, c and s must be rebuilt. Monitoring this event is equivalent to monitoring the spatial predicates $i \ominus (s \bigoplus \delta)$ for each $i \in \{a, b, c\}$. The event is triggered when any of these predicates becomes false.

3 Safe Region

The notion of *safe region* has been explored in the area of moving object monitoring [6, 8]. We apply the same notion but provide a set of formal definitions and operations on the safe regions. To simplify the presentation, we assume that a moving object is represented by a point.

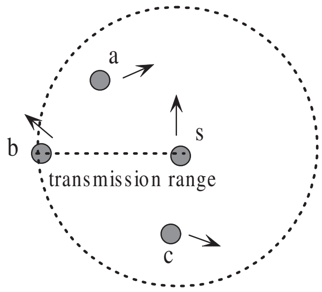
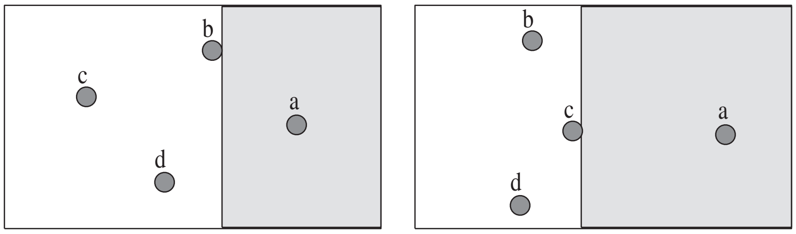


Figure 1: An Example of Monitoring Spatial Events in A Mobile Ad-Hoc Network

Definition 2: The *safe region* of a moving object a at time t_0 with respect to the spatial predicate $a \circ b$, denoted as $SR(a \circ b, t_0)$, is defined as the largest connected region within which $a \circ b$ does not change its value, i.e., $\forall s \in SR(a \circ b, t_0)$, $s \circ b = a(t_0) \circ b$, where $a(t_0)$ denotes the location of a at time t_0.

The movement of b or other moving objects may change $SR(a \circ b, t_0)$ (e.g., another object becomes "nearest to" b). Figures 2(a) and (b) show the safe regions (the shadowed areas) of object a with respect to the spatial predicate "easternmost" at time t_0 and t_1.



(a) At Time t_0

(b) At Time t_1

Figure 2: $SR(a \circ E)$ at t_0 and t_1

3.1 Operations on Safe Regions

In this subsection, we introduce the set of operations on safe regions, namely, computation, approximation, and combination. In the sequel, we focus on the "within" ($\ominus$) predicate only, but the same rationale applies to the other predicates.

3.1.1 Computation

For $a \ominus b$, if the observer b is a stationary region, $SR(a \ominus b, t_0) = b$ if $a(t_0) \ominus b = true$; otherwise $SR(a \ominus b, t_0) = D - b$, where D is the universe of the space. However, if b moves over time, $SR(a \ominus b, t_0)$ is a function of b's location. To decouple its dependence on b's movement, let ϕ_b denote the maximum velocity of b. We have the following theorem:

Theorem 3: If $a(t_0) \ominus b(t_0) = true$, $SR(a \ominus b, t_0) \subseteq b(t_0) \boxminus \mathcal{C}_{\phi_b \cdot t}$; otherwise $SR(a \ominus b, t_0) \subseteq D - (b(t_0) \boxplus \mathcal{C}_{\phi_b \cdot t})$, where $\mathcal{C}_{\phi_b \cdot t}$ denotes the circle centered at the origin and with radius $\phi_b \cdot t$, and $\boxminus$ is the Minkowski subtraction.

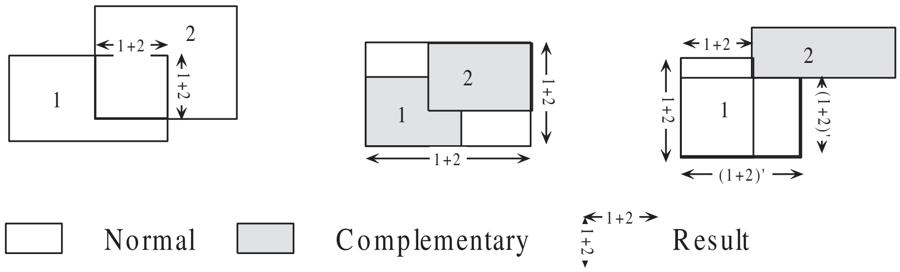


Figure 3: Three Cases of Combining Two Bounded Boxes

3.1.2 Approximation

To reduce the storage and computational cost of an arbitrary-shaped safe region, we approximate the arbitrary shape with a shrinking rectangle (or the complement of an expanding rectangle) with respect to the elapsed time t. The rectangle is called the *bounded box* of the safe region, and is denoted by $BB(a \circ b, t_0)$. At any time t, $BB(a \circ b, t_0) \subseteq SR(a \circ b, t_0)$. The bounded box is organized as follows:

Definition 4: Let the 4-tuple $\langle x_{low}, x_{high}, y_{low}, y_{high} \rangle$ denote a rectangle $\mathbf{R}$, with the tuple elements representing the four bounding edges of $\mathbf{R}$. $BB(a \circ b, t_0)$ is a 3-tuple: $\langle complement, \mathbf{R}_{t_0}, \mathbf{V} \rangle$, where $\mathbf{R}_{t_0}$ is the initial bounded box at t_0, $\mathbf{V}$ is its shrinking/expanding velocity on the four edges, and $complement$ is a Boolean designating whether the bounded box is shrinking or expanding.

To compute the approximation for $BB(a \ominus b, t_0)$, we replace $\mathcal{C}$ with its minimum bounding box in Theorem 3, and then compute the Minkowski sum or subtraction.

3.1.3 Combination

The bounded boxes of two safe regions, $BB(a \circ b, t_0)$ and $BB(a \circ c, t_0)$, can be combined (denoted as "$+$") to form a new bounded box in which the movement of object a does not change the values of both spatial predicates. The combination can be extended to more than two safe regions, which is denoted as "$\vee$".

To decide the $complement$ value of the combined box, we distinguish three cases: (1) combining two normal boxes, (2) combining two complementary boxes, and (3) a normal box combining a complementary one. Figure 3 illustrates the operation. For case (1), the combination is a normal box, and its $\mathbf{R}_{t_0}$ is the intersection of the two original $\mathbf{R}_{t_0}$'s, and its $\mathbf{V}$ is the maximum of the two original $\mathbf{V}$'s. For case (2), the combination is complementary, and its $\mathbf{R}_{t_0}$ is the minimum bounding box of the two original $\mathbf{R}_{t_0}$'s, and its $\mathbf{V}$ is the maximum of the two original $\mathbf{V}$'s.[1] For case (3), the result is defined as a normal box, and its $\mathbf{R}_{t_0}$ is the difference between the normal's and the complementary's $\mathbf{R}_{t_0}$'s,[2], the speed of its $\mathbf{V}$ is the maximum speed of the two original $\mathbf{V}$'s, and the direction of its $\mathbf{V}$ is opposite to that of the original normal $\mathbf{V}$.

4 The Abstract System Model

In this section, we describe the system model under which our location update schemes are developed. The model is abstract in that it makes no assumption on the underlying network infrastructure. Figure 4 shows the relations between the mobile clients (MC), the location server (LS) and the location-aware services (LAS): the LASs register the spatial predicates at the LS; and the MCs update their locations to the LS, which reports to the LASs when the monitored values change.

[1] By maximum, we mean the velocity that has the maximum speed.

[2] If the difference is not a rectangle, $\mathbf{R}_{t_0}$ is defined as the maximum bounded box of the difference.

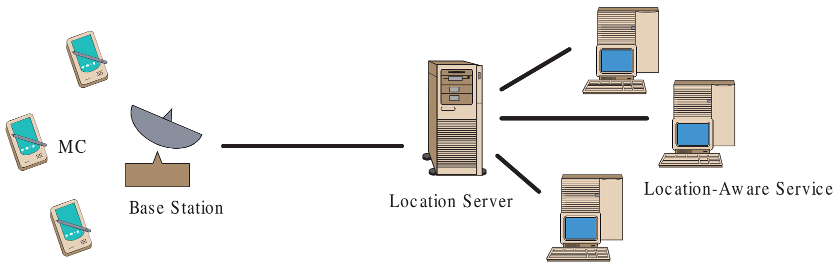


Figure 4: The Abstract System Architecture

We make the following assumptions and notations on the MC's behavior:

- The MC measures its location and determines the necessity of a location update every t_m unit of time.
- The location update is sent through an uplink channel, which costs a constant power consumption $cost_u$.
- The LS then sends to the MC through a downlink channel the approximated and combined bounded boxes for the safe regions with respect to all the spatial predicates being monitored.[3] The power consumption to receive a bounded box is $cost_d$.
- The power consumption for the MC to test if it is inside a bounded box is $cost_c$. As such, if n bounded boxes are sent to the MC, it needs $n \cdot cost_c$ in every t_m time units to make the location update decision.

5 Energy-Efficient Update Schemes

Conventional periodic or distance-based location update schemes [7] are not optimal in terms of update frequency. Better schemes should take into account the locality of the monitored spatial predicates so that the clients perform updates only when some predicate value might change. In its simplest form, the server sends back to the client the safe region of each predicate with respect to its current location so that the MC can check locally if its movement changes the value of some predicate. However, this approach incurs significant bandwidth and CPU overhead as the client has to test if it moves out of any safe region. In what follows, we present a family of energy-efficient schemes that *approximate* the safe regions with their bounded boxes, and they differ in their ways of combining these boxes.

More specifically, we first compute the safe region and then the bounded box of each predicate. All the normal bounded boxes are simply combined to form a single normal box BB_*. The proposed schemes differ in how they treat the complementary boxes. In short, a complementary box BB_c can be either discarded (if $BB_* + BB_c = BB_*$), combined with BB_*, left alone, or combined with other complementary boxes. The tradeoff is the accuracy of the approximation and the power consumption in receiving and testing these boxes. The proposed schemes aim to minimize the *average power consumption rate* before the next location update. We target the power consumption not only because this is one of the major issues in mobile and pervasive computing, but also because the reduction of location update cost in terms of power consumption also addresses other major issues such as bandwidth and response time. In the following, we first develop the power consumption model and then propose the schemes.

[3]There are three reasons why the safe region information is sent as the LS's response to the MC's location update. First, this is the only time when the LS has the exact knowledge about the MC's current location, so the safe region(s) best suits MC's need. Second, we do not assume a broadcast channel in this architecture because the power consumption of listening on this channel is forbiddenly high. Third, a request-response communication incurs less connection overhead than two independent one-way communications.

5.1 The Power Consumption Model for Mobile Clients

Let n denote the number of bounded boxes sent to the client and $\overline{t_d}$ the average elapsed time until the next location update. The average power consumption rate before the next location update, denoted as $\overline{cost_t}$, is defined as:

$$\overline{cost_t} = \frac{cost_u + n \cdot cost_d + n \cdot cost_c * \frac{\overline{t_d}}{t_m}}{\overline{t_d}} = \frac{cost_u + n \cdot cost_d}{\overline{t_d}} + \frac{n \cdot cost_c}{t_m}, \tag{1}$$

where t_m, $cost_c$, $cost_d$ and $cost_u$ are constants as defined in Section 4.

To derive $\overline{t_d}$, we make the following assumptions on the objects' movement:

- The object can freely choose a random moving direction and speed.
- It does not change the direction and speed until the next location update.
- The average speed is $\overline{\phi}$.

In other words, a moves along a straight line starting from its current updated location $a(t_0)$. On the other hand, $\overline{t_d}$ also depends on the bounded boxes, which change with time. Thus, at any given time t, the average elapsed time until the next update $\overline{t_d}(t)$ is:

$$\overline{t_d}(t) = \frac{\int_0^{2\pi} |kd_t(\theta)a(t)|d\theta}{2\pi\overline{\phi}}, \tag{2}$$

where θ is the angle between the moving direction and the positive x-axis, kd_t is the first intersection point of this line with the boundary of $BB_*(t)$ and all $BB_c(t)$. Obviously $\overline{t_d}$ is a solution to the following equation of t:

$$t = \overline{t_d}(t) \tag{3}$$

To solve $\overline{t_d}$ efficiently using Equation 3, we propose an approximation of $\overline{t_d}(t)$.

Lemma 5: Given a connected region R and an inner point p, if for any θ, the line from p with an angle θ to the x-axis intersects R's boundary only once (the intersection is denoted as $kd(\theta)$), then

$$\int_0^{2\pi} |kd(\theta)p|d\theta \approx \frac{2 \cdot Area(R)}{\delta(R)},$$

where $\delta(R)$ is the average distance between two points in R.

We replace p with $a(t)$ and replace R with $BB_*(t) - \vee_c BB_c(t)$ in the lemma, and obtain the following approximation of $\overline{t_d}(t)$ from Equation 2:

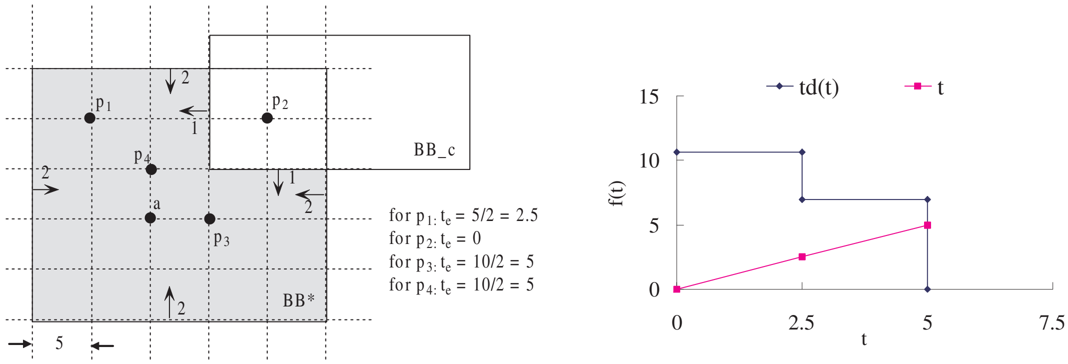
Theorem 6:

$$\overline{t_d}(t) \approx \frac{Area(BB_*(t) - \vee_c BB_c(t))}{\pi\overline{\phi} \cdot \delta(BB_*(t) - \vee_c BB_c(t))}$$

To compute $\overline{t_d}(t)$ in Theorem 6, we apply the Monte Carlo method to compute $Area$ and δ of $BB_*(t) - \vee_c BB_c(t)$. First, we randomly pick K fixed points in $BB_*.\mathbf{R}_{t_0}$ as seeds. For each seed p, let $p.t_e[i]$ denote the elapsed time from t_0 to the time when p becomes outside an individual BB_* or BB_c box i, and $p.t_e$ denote the smallest value $p.t_e[i]$ among all i. As such, $p.t_e$ is essentially the elapsed time from t_0 to the time when p leaves $BB_*(t) - \vee_c BB_c(t)$.

Each seed p accounts for an area of size $Area(BB_*.\mathbf{R}_{t_0})/K$. p contributes this portion of area to the area of $BB_*(t) - \vee_c BB_c(t)$ only when $t \le p.t_e$. Similarly, $\delta(BB_*(t) - \vee_c BB_c(t)) = 2\sum_{p,p'} |pp'|/K(K-1)$ and each p contributes to δ only when $t \le p.t_e$. As t increases, more and more p's stop contributing to $Area$ and δ. Therefore $\frac{Area(BB_*(t) - \vee_c BB_c(t))}{\pi\phi \cdot \delta(BB_*(t) - \vee_c BB_c(t))}$, i.e., $\overline{t_d}(t)$, is a staircase function. As $\overline{t_d}$ is the solution of Equation 3, it is the intersection point of functions $\overline{t_d}(t)$ and t.

An example of this method is illustrated in Figure 5. Figure 5(a) shows the BB_*, BB_c and the four seeds p_1 through p_4, and the grey area is $BB_*(t_0) - \vee_c BB_c(t_0)$. Figure 5(b) shows the corresponding $\overline{t_d}(t)$ function computed through Theorem 6. $\overline{t_d}$ is its intersection point with function t, which is 5.



(a) The Setting

(b) The Staircase Area Function

Figure 5: The Monte Carlo Method to Compute $\overline{t_d}$

5.2 The Greedy Split Scheme (GSS)

From Equation 1, minimizing $\overline{cost_t}$ is equivalent to finding a tradeoff between n, the number of BB_c's sent to the client and $\overline{t_d}$. At one extreme, all BB_c's can be sent individually (i.e., maximizing n) while at the other extreme, all of them can be combined into one BB_c (i.e., minimizing n). The family of update schemes takes a moderate approach and divides BB_c's into groups. Each group of BB_c's is combined into one BB_c and sent to the client. In this and the next subsections, we present three of the family members with different grouping algorithms.

The GSS scheme initially considers all BB_c's as one group and greedily divides it into subgroups. The scheme applies the nodes splitting algorithms in R-tree or R*-tree [5, 1]. The splitting of a group continues until it contains only one box or a further split cannot reduce $\overline{cost_t}$.

As for the time complexity, each splitting takes linear or quadratic time (depending on the splitting algorithm) of C, the number of BB_c's. It also requires $O(K)$ to compute the $\overline{t_d}$ with an incremental Monte Carlo method. As the maximum number of splitting is at most $C-1$, the overall time complexity of the GSS scheme is $O(C(C+K))$ for the linear splitting algorithm and $O(C(C^2+K))$ for the quadratic algorithm.

5.3 The Greedy Sort–Merge Scheme (GSM)

The GSM scheme first sorts all the BB_c's by some "Space Filling Curve" [2] according to their spatial locality. Then it linearly scans from the first box ($BB_c[0]$) to test if it should combine with the next box ($BB_c[1]$). The two boxes are combined if the $\overline{cost_t}$ for the combined box is smaller; otherwise $BB_c[1]$ starts a new group.

As for time complexity, the initial Monte Carlo method to compute $\overline{t_d}$ takes $O(K(C+K))$ time. Further, each test on combining involves an incremental Monte Carlo method that costs $O(K)$ time. Since the maximum number of such tests is $C-1$, the overall time complexity of the GSM scheme is $O(K(C+K))$.

5.4 The p-Means Clustering Scheme (PMC)

The p-Means clustering scheme (PMC) applies a clustering approach: it first chooses some centroid BB_c's as the initial clusters. Then it tests each of the remaining BB_c's if it should be combined with the nearest BB_c cluster or should become a new cluster. The initial set of clusters should be chosen to be the dominant part of $\overline{t_d}$ so that the scheme is stable. From the Monte Carlo method, the more p's whose $p.t_e$ values are determined by a

BB_c, the more significant this BB_c is in contributing to $\overline{t_d}$. As such, we denote the *significance* of a BB_c as the number of such p's and the initial BB_c clusters are the most significant boxes.

As for time complexity, the selection of the initial BB_c clusters takes $O(K(C + K) + ClogC)$ time. In the worst case, the test on each of the remaining BB_c's needs $O(C + K)$ time. Therefore, the overall time complexity is $O(C(C + K) + K^2)$.

6 Conclusion and Vision

To minimize the power consumption cost of location updates, we proposed a family of update schemes for spatial predicate monitoring. Our contributions are:

1. We proposed the problem of spatial predicate monitoring and showed its correspondence to spatial query monitoring.
2. We developed a precise cost model of power consumption for the monitoring problem.
3. We proposed a family of location update schemes that reduce the power consumption based on the cost model.

We foresee the ever-increasing demand of moving object monitoring, and in particular monitoring techniques that can tolerate uncertain or imprecise results. The tradeoff between the precision and the cost of the monitoring is critical in mobile ad-hoc and sensor networks and calls for more research efforts. We also envision the mass deployment of dedicated (moving) location sensors that can form ad-hoc networks and monitor other moving objects. This leads to a more challenging monitoring problem in which various factors such as the sensor movement, the sensing frequency and the update frequency should be considered.

References

[1] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *ACM SIGMOD Conference*, pages 322–331, 1990.

[2] T. Bially. Space-filling curves: Their generation and their application to bandwidth reduction. *IEEE Transactions on Information Theory*, 15(6):658–664, 1969.

[3] Open Geospatial Consortium. OGC reference model. http://www.opengeospatial.org/specs/?page=orm.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997.

[5] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD Conference*, pages 47–57, 1984.

[6] H. Hu, J. Xu, and D. Lee. A generic framework for monitoring continuous spatial queries over moving objects. In *ACM SIGMOD Conference, Baltimore, Maryland*, pages 479–490, 2005.

[7] V. Kumar and S. R. Das. Performance of dead reckoning-based location service for mobile ad hoc networks. *Wireless Communications and Mobile Computing Journal*, 4(2), March 2004.

[8] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *IEEE Transactions on Computers*, 51(10):1124–1140, 2002.

Spatial Expressions and Rules for Location-based Services in Oracle

Aravind Yalamanchi, Ravi Kothuri, and Siva Ravada
Oracle Corporation, Nashua NH 03063
{aravind.yalamanchi, ravi.kothuri, siva.ravada}@oracle.com

Abstract

In this paper, we present Oracle technologies for handling spatial events and specifying arbitrary spatial rules. The framework allows construction of arbitrary expressions involving spatial predicates, indexing them using an expression filter index, and incorporating them as conditions within an event-condition-action (ECA) rule. This framework provides a seamless integration of multiple components within Oracle and provides a rich set of functionality for location-based service applications.

1 Introduction

Recent advances in mobile positioning technologies such as GPS, and A-GPS have simplified and reduced the cost of automatic location acquisition thereby enhancing the use of location in a wide variety of applications. Examples of such applications include location-based services, routing and tracking of shipments, luggage or similar items, and supply chain management using RFID. Of these, location-based services (LBS) use location explicitly. Simple examples include geocoding, mapping, and routing (driving directions) [HB04]. Advanced applications combine positioning and location-based analysis with an event alerting/management systems. [JC+04] lists several categories of LBS applications such as traffic coordination in the presence of bottlenecks, safety-related services and location-aware content delivery services. Specific examples in the last two categories are:

- Secure-Zone Monitoring: Secure regions around important places of interest could be defined using spatial buffering functions. These regions can later be used in screening flights entering those regions for security violations.
- Car Buyer Subscription Service: An interested buyer can register for a service and specify preferences for his ideal car using attributes of the car (model, price, etc.) and the distance he will travel to obtain the car. When a car enters the market, the potential list of buyers whose preferences are met by the car being sold is notified.

In order to enable the above types of location-based service applications, event management functionality capable of handling spatial events and preferences defined on such spatial events is required. In this paper, we describe a framework for managing spatial events inside an Oracle database. This framework can be easily deployed in a variety of location-based service applications including the ones described above.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Spatial Event Management in Oracle 10g

To support location-based service applications, Oracle provides a generic framework as illustrated in Figure 1. The framework consists of the Rules Manager component, the Spatial and the Expression filter engines, a visualization component called Mapviewer and an alerting component.

The Rules Manager allows users to define rules inside an Oracle database. A rule is modeled using *event*, *condition*, and *action* semantics [WC96]. The spatial and expression filter components enable the condition part of a rule, and the alerting component enables the action part of a rule. Events triggering the rule can be regular database events such as triggers. Some of these events can be triggered due to the positioning aspect of a mobile user. The Oracle Application Server (Wireless) component supports GPS positioning and network-based positioning interfaces such as MLP, ParlayX provided by mobile operators. Location-based service applications can make use of these different technologies available in the Oracle Database and Application Servers. In this paper we describe the overall framework for managing spatial events using the spatial, expression filter and rule manager components of the Oracle Database Server. In the next few sections, we describe each of these components in more detail.

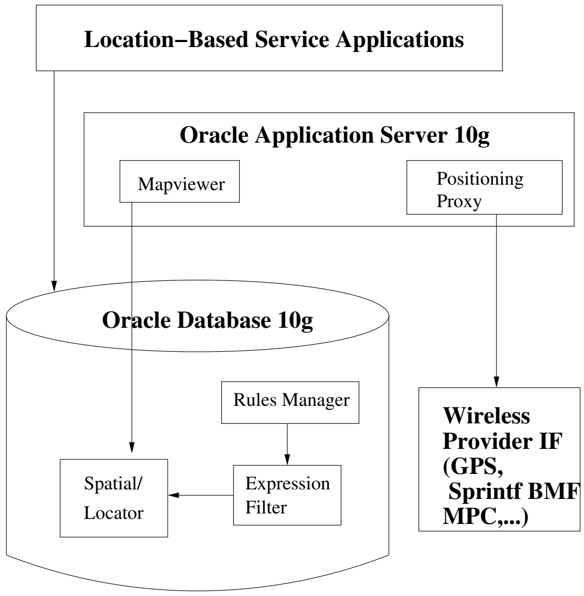


Figure 1: Framework for Spatial Events in Oracle.

3 Oracle Spatial

In most applications, location-based services heavily rely on GIS-type functionality. The Spatial engine of Oracle provides the required GIS functionality for location-based services. The functionality includes:

Data Model for Location data: Spatial defines a data type called SDO_GEOMETRY to store locations of data items. Users can create tables with columns of type SDO_GEOMETRY and index such columns using specialized indexes such as R-trees, or Quadtrees [KRA02]. Such indexes facilitate fast responses to location-based queries.

Query and Analysis: Relational operators such as $<$ are used to compare numeric or string attributes of a table. For the SDO_GEOMETRY columns, Spatial provides specialized operators to perform location-based filtering, query, and analysis. Commonly used operators include (but not limited to) the following:

- SDO_WITHIN_DISTANCE: This operator identifies all rows of a table where the corresponding location data are within a specified distance of a query location. This operator can be used to identify the restaurants in 2-mile radius of a user's current location.
- SDO_RELATE: This operator identifies all rows of a table where the corresponding location data interacts in a specified way with a query region. The types of interaction allowed with this operator include - inside, contains, covers, and anyinteract. If a mobile user has a preferred region of interest, this operator can be used to identify the restaurants in this preferred region.
- Others: OGC and SQL/MM recommend a list of other spatial functions. A majority of these functions/operators are supported by Oracle Spatial and are listed in detail in the documentation [Ora10g].

Geocoding: Oracle Spatial provides functions to convert user's addresses to geographic locations (usually specified as <longitude, latitude> coordinates). Such conversion is called geocoding in GIS parlance. Reference data used in geocoding is obtained from third-party vendors such as NAVTEQ.

Routing: Spatial provides APIs to compute routes between a starting and an ending location. The route includes driving directions from the starting location to the ending location.

4 Expression Filter

Traditional databases support storing data values in relational tables and finding interesting data values using a *conditional expression* in the WHERE clause of a SQL query. Recently, an inverse paradigm has generated considerable interest in the database community. This paradigm stores the conditional expressions in a relational table and these expressions are matched with a transient data item specified in a SQL query [YSG03]. The Expression Filter feature [OraA10g] in Oracle database provides appropriate functionality in this paradigm. For instance, in a Car-Buyer Subscription Service, Buyers' interests in cars can be captured as conditional expressions that are stored in a column of Expression data type in a relational table(Table 1).

Customer_id	Zipcode	..	Interest
1	32611	..	Model='Taurus' and Price $<$ 15000 and Mileage $<$ 25000
2	03060	..	Model='Mustang' and Year $>$ 1999 and Price $<$ 20000
..	..	..	..

Table 1: Customer table specifying conditional expressions in the *Interest* column.

Each expression column is associated with an object type that defines necessary metadata (attributes used in the expressions and their data types) for the expressions. For the previous application, a Car4Sale object type with Model, Mileage, Year, and Price attributes is configured as the metadata for the expression column.

The Car-Buyer Subscription Service can be modeled to monitor cars as they go on sale. For this purpose, a car, represented as an instance of the Car4Sale object type, can be matched with the expressions in the Interest column of the Consumer table using the following query.

```
SELECT customer_id FROM Customer
  WHERE EVALUATE(Customer.interest,
               AnyData.convertObject
                 (Car4Sale('Mustang', 19000, 25000, 2001))) = 1;
```

The EVALUATE operator, provided by the Expression Filter feature, facilitates evaluation of conditional expressions (stored in the *Interest* column) for a data item using standard SQL statements. A specialized index can be defined on the column storing expressions to speed up the above query.

```
CREATE INDEX cstidx ON Customers(interest) INDEXTYPE IS EXFSYS.EXPFILTER;
```

For the purpose of indexing expressions, the predicates in the expression set are grouped based on the commonalities in the predicates and stored in a relational table called the predicate table [YSG03]. A subset of the predicate groups are identified as *indexed predicate groups* based on the frequency and selectivity of corresponding predicates. Indexes are used for the evaluation of such predicate groups. Non-indexed predicate groups are processed in the final stage of expression evaluation.

4.1 Spatial Predicates in Expression Columns

In the Car-Buyer Subscription Service, the buyer may often have a preference on the spatial proximity of the car in addition to the traditional attributes (Model, Mileage etc.,). To support such applications, the metadata associated with the Expression column may be extended to include an attribute, VehLoc, that captures the location data for the car going on sale. Using this attribute, the customer may now restrict his search to a region, for example, a 2-mile radius of his current location (say longitude -79, latitude -38). This is possible by allowing a mix of scalar predicates using operators such as $<$ (as in Price $<$ 15000) and spatial predicates using SDO_WITHIN_DISTANCE and SDO_RELATE operators in the stored expressions. The Table 2 [1] shows one such example.

Customer_id	Zipcode	..	Interest
1	32611	..	Model='Taurus' and Price < 15000 and Mileage < 25000 and **SDO_WITHIN_DISTANCE(VehLoc, POINT(-79, -38)**, **'distance=2 unit=MILE')='TRUE'**
2	03060	..	Model='Mustang' and Year > 1999 and Price < 20000
..	..	..	..

Table 2: Customer table using spatial operator in the *Interest* column.

When an index is defined on the Interest column, all the spatial predicates form an *indexed predicate group* and the predicate table structure is extended to include an SDO_GEOMETRY column to store the *spatial regions* that are computed from these predicates. For the previous example with SDO_WITHIN_DISTANCE operator, a 2-mile buffer around the specified location is computed and stored in this column. In the case of a spatial predicate involving SDO_RELATE operator, the location specified in the predicate is stored in this column and the exact spatial interaction (inside, contains, touches etc.,) to be matched is captured as a function that is processed in the final stage of expression evaluation. A spatial index is defined on the column storing spatial regions in the predicate table. In order to evaluate the spatial predicates, the location data from the data item passed to the EVALUATE operator is matched with the *spatial regions* in the predicate table for any interaction [2]. For the expressions matching the location data (and also matching on all other indexed predicate groups), the remaining predicates associated with the expressions are evaluated for conclusive matches.

When a new car with an appropriate location enters the market, an instance of the Car4Sale object type, which now includes the VehLoc attribute for capturing the vehicle location, is passed to the EVALUATE operator as shown below.

[1] In this table, POINT is a function that takes longitude, latitude coordinates and returns an SDO_GEOMETRY.

[2] Note that SDO_WITHIN_DISTANCE operator is equivalent to an any interact operation on computed buffer

```
SELECT customer_id FROM Customer
  WHERE EVALUATE(Customer.interest,
                 AnyData.convertObject
                   (Car4Sale('Mustang', 19000, 25000, 2001, POINT(-79, -37.99)))
                 ) = 1;
```

Note that the POINT(-79, -37.9999) function captures the location of the car. This query uses the index defined on the Interest column to evaluate the expressions based on spatial and non-spatial predicates. The expression filter index in turn uses the spatial index defined on the column storing spatial buffers in the predicate table and the bitmap indexes defined for the other indexed predicate groups together to filter the expressions based on multiple predicate groups simultaneously. [3]

In order to provide alerting service for matching customers, the results from the above query can be used as input to other database components supporting notification framework. For example, if the consumer table includes the email address of each consumer, the UTL_SMTP package in Oracle database can be used to send out alerts as email messages to the matching customers. Alternately, if the matching of a car to a customer's interest should trigger a workflow, Advance Queues [Ora10g] in the Oracle database may be used.

5 Rules Manager

The latest release of Oracle RDBMS provides the Rules Manager framework to define and enforce *Event-Condition-Action(ECA)* rules [WC96] in the database. The expressions described in the previous section form the basis for the *condition* part of a rule. In this section, we will focus on the remaining aspects of managing rules such as *events* and *actions*.

An *event* is the occurrence of some state change in a component of a software system, made visible to the external world [ON03]. An event is an instance of a data structure, called the *event structure*. The event structure provides the necessary vocabulary for the rule condition by specifying the list of attributes and their data types. A rule condition defined using the attributes from an event structure evaluates to true or false for an instance of the event structure (event). The *action* associated with the rule is a program that executes in the host application when the corresponding rule condition evaluates to true.

The rules supported by the Rules Manager can be represented using standard ECA notation [Han92], where the ON clause identifies the event structure for which the rule is defined, the IF clause identifies the rule condition and the THEN clause identifies the rule action. The following code sample shows two rules defined for the Car4Sale event structure.

```
ON   Car4Sale(Id, Model, Mileage, Year, Price, VehLoc) car
IF   Model = 'Taurus' and Price < 15000 and Mileage < 25000 and
      SDO_WITHIN_DISTANCE(VehLoc, POINT(-79,-38)
                          'distance=2 unit=MILE') = 'TRUE'
THEN SendCarInfo('foo@abc.com', car);

ON   Car4Sale(Id, Model, Mileage, Year, Price, VehLoc) car
IF   Model = 'Mustang' and Year > 1999 and Price < 20000
THEN SendCarInfo('bar@abc.com', car);
```

In the database, the event structure (Car4Sale) is captured as an object type with the complete set of attributes. Each instance of this object type (event) can be explicitly added to the Rules Manager to evaluate the corresponding rules. In a rule based application, multiple rules may be defined using the same event structure. These rules together form a logical set called the rule class and they represent one unit of evaluation. That is,

[3] See Oracle documentation for details on multi-dimension filtering using index predicate groups.

for each instance of the event structure (event), the rules in a rule class are all evaluated and the actions for the rules matching the event are executed.

The rule actions are carried using a callback procedure, which at the time of action execution has access to the event instance and the rule it matched. Hence the attributes that determine the appropriate action for each rule, such as the email address of the subscriber, are associated with the rules as action preferences.

At the time of rule class creation, an association is established between the event structure, the action callback procedure and a rule class table that acts as the repository for rules. The following commands are used to create a rule class for 'CustomerRules' application involving a 'Car4Sale' event structure and a 'CustomerAlerts' action callback.

```
BEGIN
  dbms_rlmgr.create_rule_class(rule_class_nm => 'CustomerRules',
                               event_struct  => 'Car4Sale',
                               action_cbk    => 'CustomerAlerts');
END;
```

rlm$ruleid	EmailAddress	..	rlm$rulecondition
1	foo@abc.com	..	Model='Taurus' and Price $<$ 15000 and Mileage $<$ 25000 and SDO_WITHIN_DISTANCE(VehLoc, POINT(-79, -38), 'distance=2 unit=MILE')='TRUE'
2	bar@abc.com	..	Model='Mustang' and Year $>$ 1999 and Price $<$ 20000
..	..	..	..

Table 3: CustomerRules rule class table with rule conditions and action preferences.

In a database schema, a rule class is represented as a relational table with an Expression column that stores the rule conditions (See Table 3). The event structure for the rule class is used as metadata for the expression column. Additionally, it has one or more columns to store the action preferences for each rule. The values stored in these columns are used to determine the appropriate action for each rule as shown in the following implementation of the action callback procedure.

```
PROCEDURE CustomerAlerts(car       Car4Sale,
                         rlm$rule  CustomerRules%ROWTYPE) IS
BEGIN
  SendCarInfo(rlm$rule.EmailAddress, car);
END;
```

The events (instances of Car4Sale object type) are submitted to the rule class for evaluation using a programmatic interface. This step processes the rule condition and invokes the action callback procedure for each matching rule.

```
BEGIN
  DBMS_RLMGR.PROCESS_RULES(
      rule_class_nm => 'CustomerRules',
      event_inst => AnyData.convertObject
                     (Car4Sale('Mustang', 19000, 25000, 2001,
                               POINT(-79, -37.9999))));
END;
```

5.1 Applications involving Composite Events

Composite events [CKAK94] are formed out of two or more simple (primitive) events using an event pattern language with correlation operators such as conjunction, disjunction, negation, and sequence. The simple or primitive events that constitute a composite event may occur in different applications and at different times.

The Rules Manager framework can be used to support rule applications dealing with composite events. In this case, the rule condition can be expressed to span multiple events that occur independent of each other. The action for a rule is executed when all the necessary events are processed. In this scenario, each rule condition acts as a state machine with possible intermediate states and the rule condition is considered true when the state machine reaches an accepting state. For example, a consumer may express interest in a car based on the car attributes as well as the vehicle history report that is independently obtained for each car. One such rule can be expressed using an XML and SQL based rule condition language as shown below.

```
ON Car4Sale(Id, Model, Mileage, Year, Price, VehLoc) car
   VehHist (CarId, Rental, FireDamage, InsuranceLoss ) hist
IF
   <condition>
     <and join="car.Id = hist.CarId">
       <object name="car">
          Model = 'Taurus' and Price < 15000 ...
       </object>
       <object name="hist">
          FireDamage = 'NO' and InsuranceLoss < 5000
       </object>
     </and>
   </condition>
THEN SendCarInfo('foo@abc.com', car);
```

For the above rule condition, when a Car4Sale event occurs, the first half of the rule condition is matched and the intermediate results are stored in the database. Later, when a VehHist event for the same car occurs, it matches the second half of the rule condition and satisfies the join condition with the Car4Sale event in the event history, resulting in the execution of the rule action.

The complete XML and SQL based rule condition language is capable of handling most common event correlations such as sequence of events, conjunction of events, disjunction of events, and negation of events. This enhanced language can be used to correlate multiple spatial or non-spatial events in a uniform manner. For additional information on the rule condition language and the event management policies that dictate the behavior of the events in the system see [OraA10g].

6 Conclusions

In this paper, we described a framework for managing spatial events in an Oracle database. This framework allows combining of spatial events with non-spatial events using ECA rule semantics. The conditions in the framework are indexed using expression filter and the underlying spatial indexes thereby scaling the system to millions of rules. This integrated solution using expression filter, spatial, and alerting components in Oracle extend the salient features of commercial databases such as backup and recovery, concurrency, and replication to location-based service applications.

References

[CKAK94] S. Chakravarthy, S. V. Krishnaprasad, E. Anwar, and S. K. Kim. Composite Events for Active Databases: Semantics, Contexts and Detection. In *20th International Conference on Very Large Databases Conference*, 1994.

[Han92] E. Hanson. Rule Condition testing and action execution in Ariel. In SIGMOD Conference, pages 49-58, 1992.

[HB04] M. Horhammer and P. Biswas. A Rule Engine for Location Based Content Syndication. In *IEEE Intl. Conf. on Mobile Data Management*, 2004.

[HJ04] X. Huang, and C.S. Jensen. Towards A Streams-Based Framework for Defining Location-Based Queries. In *STDBM*, pages 73–80, 2004.

[J04] C.S. Jensen. Database Aspects of Location-Based Services. In *Location-Based Services*, pages 115–148, 2004.

[JC+04] C.S. Jensen, A.F. Christensen, T. B. Pedersen, D. Pfoser, S. Saltenis, and N. Tryfona. Location-Based Services – A Database Perspective. In *ScanGIS*, pages 59–68, 2001.

[KRA02] Ravikanth V. Kothuri, S. Ravada, and D. Abugov. Quadtree and R-trees in Oracle: A Comparison using GIS Data. In *ACM SIGMOD Intl. Conference on Management of Data*, pages 546–557, 2002.

[ON03] K. S. Ozgur and E. Nadia. RUBCES : A Rule Based Composite Event System. In *International XII. Turkish Symposium on Artificial Intelligence and Neural Networks*, 2003.

[OraA10g] Oracle 10g Application Developers's Guide - Rules Manager and Expression Filter, Release 10.2. *Oracle Corporation, Part# B14288-01*, December 2003.

[Ora10g] Oracle 10g Database User's Guide, Release 10.1. *Oracle Corporation*, December 2003.

[WC96] J. Widom, and S. Ceri. Active Database Systems. In *Morgan-Kaufmann*, ISBN 1-55860-304-2, 1996.

[YSG03] A. Yalamanchi, J. Srinivasan, and D. Gawlick. Managing Expressions as Data in Relational Database Systems. In *CIDR*, pages 303–313, 2003.

Data Store Issues for Location-Based Services

John Krumm Steve Shafer
Microsoft Research
Redmond, WA 98052
{jckrumm, stevensh}@microsoft.com

Abstract

Location-based services (LBS) provide resources or information based on the user's own location or an indicated location of interest. This paper introduces a number of data store and access requirements for LBS based on examples from our own work and that of others in the field. We consider specifically the areas of location representation, gathering location data, assets, location-based queries, and location system architecture, all of which are important to location-based systems and are extensions and specializations of current database issues.

1 Introduction

Location-based services (LBS) provide resources or information based on the user's own location or an indicated location of interest. For example, an automobile's navigation system may display the locations of nearby gas stations when the car's tank is getting low, a cell phone may switch to vibrate when a person enters a theater, or a document may be directed to print near the location of a person's next meeting. A data store of some sort is needed to represent the locations in the world, as well as their attributes and relationships, and the resources available. This data store is used for interpreting sensor readings, performing spatial queries and inferences, and triggering actions. In geographic information systems (GIS), the data store is usually a geospatial database; in many indoor ubiquitous computing systems, the data store may be a simple as a drawing file. In any case, location-based services have requirements that challenge traditional data representation systems. This paper introduces a number of these requirements based on examples from our own work and that of others in the field. We consider specifically the areas of location representation, gathering location data, assets, location-based queries, and location system architecture, all of which are important to location-based systems and are extensions and specializations of current database issues.

2 Location Representation

Geometry and Objects: All location-based systems need to represent locations. The representation can be as simple as a few items in a list or as complex as a full geographic information system. Two of the more natural representations of location are metric and object-oriented. A metric representation consists of numerical coordinates (e.g. (x,y,z) and (latitude, longitude)) to specify points and shapes. In our EasyLiving project [BMK+00, BS01],

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

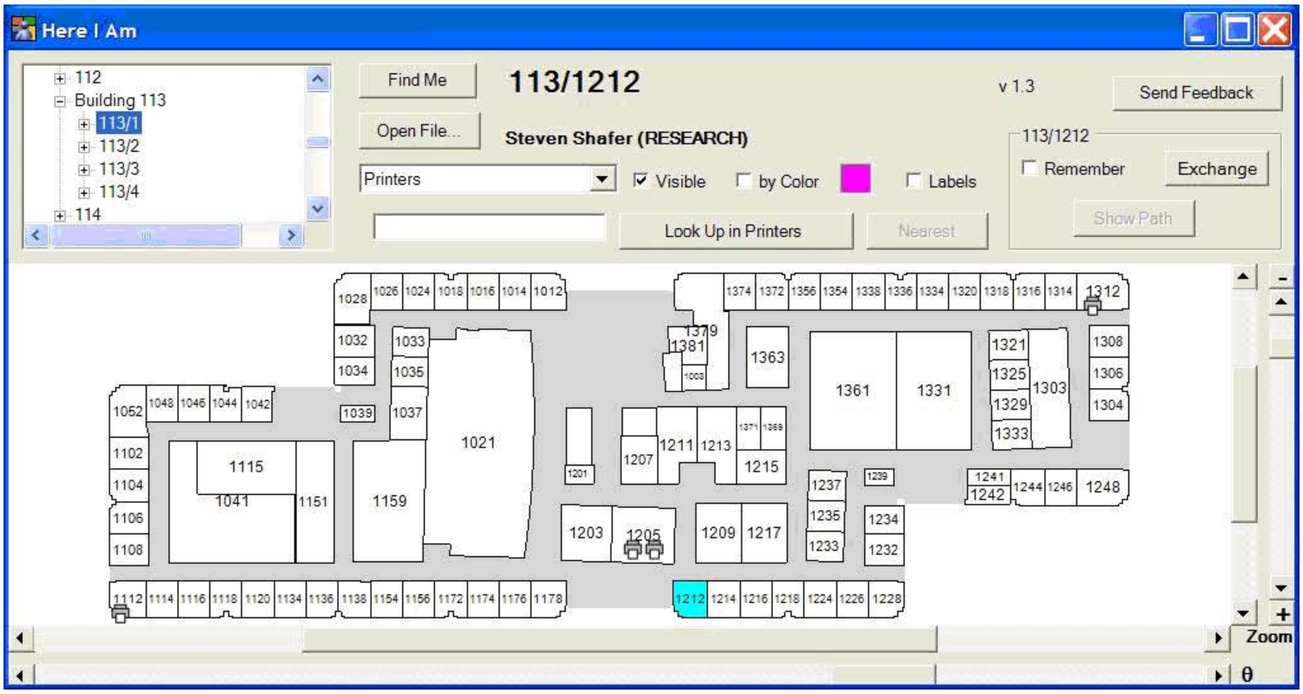


Figure 1: Our "HereIAm" program displays floor plans and contents of rooms. It represents spaces with coordinates and containment relationships.

we used a metric representation to give the locations of devices, furniture, and people on a 2D grid on the floor. This made it easy to compute when, for instance, a person passed within a certain distance of a computer screen. A metric representation may have to support multiple coordinate systems and transformation between them.

A common alternative to a metric representation is an object-oriented representation which, at its simplest, is just a list of locations, such as "kitchen", "living room", "bedroom". An object-oriented representation makes it easy to specify containment relationships in a hierarchy, such as the fact that a number of different offices are in a given wing of a building, which is on a given floor, which is in a given building. Representing spaces as objects, however, leaves out critical data for computing metric relationships like distance. This can be fixed with a hybrid approach that represents spaces with both coordinates and objects. As an example, we built the "HereIAm" program, shown in Figure 1, to represent building floor plans. HereIAm uses a hierarchical representation to relate the spaces on a given floor, the floors of a given building, the buildings on a given campus, etc. It uses a metric representation to draw all the spaces to scale. Some systems that use objects have a type definition for various types of location objects (building, floor, room, conference room, etc.) - in HereIAm, the appearance attributes of a room were given in such a type definition, but these attributes could be overridden for individual objects, for example to highlight the currently selected room in blue in Figure 1.

The Third Dimension: Another key issue is the representation of the third dimension. In many systems, the third dimension is completely absent, for example a mapping program that treats the Earth's surface as a spheroid. When the third dimension is introduced outdoors, complexities arise from the need for GPS (or other location sensor) to have more points of reference in order to determine the third dimension; the lack of standardization of the unit of height in standard outdoor coordinate system definitions; and of course the storage and rendering complexity of 3D polyhedra v. 2D polygons. Many systems do record and/or display true 3D, for example game worlds and topographic maps. Even modern aerial photograph systems are beginning to map the images onto a 3D world map with elevation, such as Google Earth (`http://earth.google.com`). Indoors, a single floorplan may be naturally represented as a 2D drawing, and this is satisfactory for many purposes. 3D indoor models are much more complex, typically used for the construction industry but not so much for consumer applications. The complexity of 3D is frequently not warranted for a given application. However, there are other choices that capture some of the elevation information such as "2-1/2 D" geometry. The definition of this term is idiosyncratic, but an example would be to store space definitions as 2D polygons in XY, with the Z coordinate a simple ordinal quantity (1 = first floor, 2 = second floor, etc.) Our EasyLiving system worked entirely in 2D, though HereIAm had a simple version of 2-1/2 D as presented here.

Associated Data: Frequently, a location data store records more than just the location descriptions, it records additional data associated with the locations. One kind of associated data is properties or attributes of each location, such as whether it is a space or a route segment; or whether it is displayed by outline, displayed by coloring its extent, or not displayed at all. Attributes can also describe how a space is used, how it is classified for tax or rental purposes, who is responsible for administering it, what its name is, and so on. Space attributes can be grouped, and sometimes this grouping forms a "type" system with spaces of each type sharing certain attributes. The type system can even allow subtyping with inheritance of attributes. Another kind of associated data found in location systems is relationships between locations, such as containment or adjacency. Storing such relationships makes it more efficient to search through the location data store, but the relationships must be computed and stored. One of the beauties of geometric coordinates is that a great many relationships - including containment and adjacency - can be computed dynamically, without the need to pre-compute and store them in the data store explicitly. However, performing such computations at run-time can be expensive - potentially extremely expensive. So, a design engineering process is needed to decide what relations to record explicitly in the data store, and what relations can be calculated dynamically.

Uncertainty: Many times location data comes from sources that are not completely reliable, such as the video cameras we used in EasyLiving [KHM$^+$00] or the Wi-Fi access points we used in LOCADIO [KH04b] to track the locations of people and devices in our space. In such cases, it is important to explicitly represent the probabilistic nature of the measured locations to avoid hiding the inherent uncertainty from down stream processes. Uncertainties can be represented as parameterized distributions like Gaussians, which are easy and compact to represent. Or they can be represented nonparametrically as, say, a cloud of particles [DdFG01], which is less efficient but more expressive.

Federation: No one database will contain all location data for all locations. Different databases will cover different geography as well as different objects. For instance, an enterprise's buildings and campuses may be represented in different files as we did in HereIAm. Different authorities may control data for different types of assets: the IT department may maintain the locations of printers and wireless access points, while the facilities department may maintain floor plans. In these cases it is important that the representation allow for the federation of different databases, perhaps by supporting pointers to data maintained by other authorities. When federation occurs across different representation systems, some interoperability standard is needed to allow locations in one system to point to locations in another. The goal of federation is to implement functional computations smoothly, as if disparate location data stores were parts of a single "virtual" unified data store.

3 Generating Location Data

Authoring and Editing: Location data can come from many sources and go to many destinations. For inputting primarily static data, such as floor plans and geographic entities, a one-time conversion into the chosen representation is sufficient, if sometimes tedious. Once in the database, location data sometimes needs manual editing to stay up-to-date. Instead of making the human editor work with traditional database update statements, it is much more practical to use a friendly, graphical front end to the database so editing becomes more like running a drawing program. When editing an existing location data store, a common approach is to update the source data (e.g. architectural drawing), and rerun the analysis program that creates the location data store representation. However, this can lead to changing the internal ID numbers associated with locations, which may be used by applications. Maintaining location ID across edits is a key problem. Similarly, it may be desirable to maintain a history of ID and other changes as an attribute of locations (current or past), such as when a wall is added or removed from a structure and the room numbers are reassigned accordingly. HereIAm uses drawings as source data; the room numbers are assigned by hand in the source drawing, and the internal IDs are derived from building names and room numbers for continuity across floorplan modifications.

Self Maintenance: Sometimes location data can be generated from the database itself. Our NearMe system [KH04a] finds nearby people and resources based on Wi-Fi access points. As people submit sets of detected access points, the NearMe server recomputes every hour the topology of overlapping access points based on which ones have been detected together. The new Location Finder in Microsoft's Virtual Earth (`http://virtualearth.msn.com`) anonymously logs access point detections to try to infer the locations of access points it has not yet seen. In these self-maintenance schemes, it is important to guard against mistaken data, either as a result of inaccurate sensors or malicious data submissions.

Use-Based Location Definition: Self Maintenance can be extended to actually identify locations of interest based on observations of "breadcrumb" data (the history of people's movements). Whenever a region has a pattern of occupancy or movement very different from its surroundings, it may be useful to define a boundary and declare it to be a place of interest. An object can be created, and it can be entered into the location representation system. For example, the Lachesis project [HT04] automatically finds a user's frequent destinations which an experimenter can manually name, while Opporunity Knocks [PLG$^+$04] asks the user to take a picture of any place it recognizes as a frequent destination. In this way, from breadcrumb data, a network of interesting places and paths can be defined based on the actual usage of spaces - transit areas, meeting areas, resting areas, etc. Three tasks that are particularly challenging are defining the boundaries of the new space, interpreting what it is used for so its attribute values can be assigned, and giving it a name that can be advertised and used in external references.

4 Locations and Assets

Assets: One key function of location-based systems is to provide access to assets of various kinds based on location. Assets can include physical things like printers or goods in a warehouse; information such as a building directory or facts about a specific painting on the wall; services such as requesting maintenance or obtaining navigation directions; or even people whose locations may be determined by sensors. Generically, we can think of an "asset roster" as a list of assets and for each, its location or service region. However, in practice, there are many ways to represent an asset roster. In one extreme, only locations are represented, and each asset is recorded as a location just like any other. For example, a printer in a room is represented as a new location object at the appropriate place and with the appropriate relations with other locations. The attributes of this location contain the detail about the fact that it's a printer. This makes queries easy, but shoehorning all interesting facts about the world into a location data store is awkward. The other extreme is to say that there is only a "data store of things", and all locations are represented as "things", with some relations between them. This also simplifies some aspects of design; but it is awkward, for example, to perform geometric computations or route selection if your only data store is generic for all kinds of things. These extremes can be characterized as "all things are locations" or "all locations are things". A more sophisticated design distinguishes between locations in the location data store and asset rosters which provide a list of assets and their corresponding locations. HereIAm distinguishes locations from assets, and allows "click-through" on the icon of any displayed asset to view or edit its properties. However, HereIAm uses a declarative representation of asset rosters, which provides no means to propagate changes back to the source data store that generated the asset roster.

Asset Roster Representation: Asset rosters can be represented in many very different ways. The simplest presentation might be a file or database table with one row for each asset, and columns for the asset ID; for other asset information such as a name, type, and pointer to more information; and for the location of the asset. An asset roster can be a file, a table in a database, or a composite of numerous tables or files. An asset roster can be a drawing file with things drawn onto it as icons or as geometric figures. In some systems, each asset is a device that maintains its own location property, which can be queried, so the "asset roster" is implicitly the distributed set of location properties maintained by all devices. There may be many asset rosters for a given location domain, and they may be administered by different authorities - for example, the Real Estate staff of an enterprise may keep the floorplan drawing database, whereas the IT staff maintains a roster or printer locations within the buildings. Issues of Federation (see above) are particularly important for asset rosters. In addition, the jurisdictional scope of an asset roster may not match that of the location store, for example the Real Estate staff at each site may maintain the building floorplans for that site, but the printer database may span several sites.

Mobile Assets: Some assets move in the world, so their location must be queried dynamically, and may be remembered in a historical record. Location updates can come from live sensors, such as the people- and device-trackers we have developed. In these cases, the database must support programmatic, real time updates. For objects that move, it is important to represent dynamics such as velocity to answer queries related to speed and to make predictions about where an object will be. In LOCADIO we used the variance of Wi-Fi signal strengths to infer probabilistically whether or not a client was moving. Representing dynamics for location-based services starts down a slippery slope of full context representation, such as a user's current activity or a conference room's availability. Asset rosters are frequently queried by copying all or part of the roster into the client application's memory space, but for dynamic assets, this may be prohibited, requiring that locations be queried every time they are to be used by the application, or according to some schedule or plan.

Own Location: Perhaps the most challenging kind of mobile asset is people, whose locations may be determined by external sensors, by a sensor they carry, or through a computing device whose location is measurable. People move around, sometimes in relatively unpredictable ways, sometimes using spaces in ways the designers did not intend, with many variations in the significance of their location. A person's location can be viewed from the standpoint of the person or the environment. From the person's standpoint, "own location" means where the person is located right now. For example, a building directory application might use the own location to select which floor's floorplan to display (this is done in HereIAm). From the environment's standpoint, all people's locations can be collectively considered to form a single asset roster of mobile assets, i.e. people. There are deep privacy issues around the disclosure of own location to others, including the centralization of own locations into a single server or data store. In addition, some location sensing systems determine own location on each mobile device (e.g. using Wi-Fi signal strengths in LOCADIO), while others are central systems that determine the locations of all mobile devices through facilities in the infrastructure of the environment (e.g. WhereNet Corporation, `http://www.wherenet.com/pdfs/VSS.9.9.03.pdf`). In the latter case, a person or device that desires to know its own location must obtain it by querying an asset roster in the environment.

5 Location-Based Queries

Geometric Queries: Location-based services are challenging in terms of database querying because of the specialized geometric nature of the queries and because of the multitude types of triggerable events. Clearly geometry is important for spatial queries. In our EasyLiving project, one of our most frequent queries concerned whether or not a person, respresented as an (x,y), had entered a particular space, such as the area around a computer monitor. Our XRay local search program [HKH05] makes queries about points of interest inside a cone-shaped region emanating from a user's current (latitude, longitude) pointing toward wherever the user is pointing his mobile device. Queries that span coordinate systems have to take into account coordinate transformations, including possibly discovering and computing long chains of transformations to connect the relevant frames of reference. Queries involving multiple shapes are also important. As an example, the region from which an audio speaker can be heard is the intersection of its audible region in free space and the walls of the room around it. Incorporating dynamics introduces the element of time, such as a query asking when a certain velocity vector will intersect a certain region. Probabilistic representations need probabilistic queries, such as, "Show me the minimum area over which the probability of a given person's location integrates to 0.99".

Precomputation: The answers to common queries can be saved for efficiency. For instance, a common query on floor plans is to list all the spaces on a floor of a given building. Similarly for routing, commonly requested routes can be stored to avoid recomputing them. One recent example of the power of precomputation for location-based services is the new Google Maps (`http://maps.google.com/`) and Microsoft's Virtual Earth (`http://virtualearth.msn.com`). Previously, Web-based maps were generated on demand, so panning even a few pixels required a complete rerendering. With precomputed tiles, new Web map cites deliver fast, interactive panning. However, prerendering may require that the zoom levels be coarsely quantized, and map labels may often be obscured by superimposed annotations.

Privacy: With location-based services receiving more attention, location privacy is the subject of increasing concern and research. Databases can help by enabling location queries that return locations of specified resolutions. This would allow, for instance, a user to specify that he is willing to share his location down the city level, but not at any higher resolution. Likewise, it can be valuable to have a database deliver outright lies about a user's location in order to reduce the confidence of anyone spying. These techniques are sometimes called "location dithering", and representative work appears in [DK05]. Privacy can also include hiding information within the location data store, for example certain buildings in a corporation may contain sensitive resources and their floorplans might not be available for viewing by all employees.

Triggers: Some queries run in the background in order to trigger location-related events. In the EasyLiving project, our database triggered events based on the room's behavior rules, such as automatically adding a user's music play list to the room's list of playable songs whenever the user entered the room. Sometimes events trigger actions outside the database, such as redrawing a space when something or someone moves. While it is usually sufficient to recheck trigger conditions whenever the database is updated, triggers can also be conditioned on inaction. For instance, in elder care, it is important to be notified if a subject has not left his or her residence for a long period of time. Just as with editing a spatial database, authoring and editing location-based triggers would be easier with a graphical interface that helps a programmer define the spatial preconditions for an event to fire.

6 Location Computing Architecture

Location Engine Structure: A location-based service needs a body of code that understands the representation and can evaluate queries such as "find nearest thing" and "find route". This body of code can be called a "location engine". One of the key design issues in location-based services is where the location engine resides - in the client (application), or in the location data store (server), or split across the two. When the data store is a true database system, or presents a server interface, the location engine can be inside the data store. A client need not understand the location representation, it can simply send a query to the data store, whose location engine will compute the desired result. This centralized approach involves small but relatively frequent network traffic, and allows very thin client applications. However, it also requires that all location calculations for all clients share the same server to do the heavy work, and it requires connectivity to the network in order for the application to execute (unless the client has precomputed and stored the results of certain queries before becoming disconnected). Such a design also gives the client a very limited language in which to express its queries, i.e. the network interface presented by the data store. An example of this is the MapPoint Web Service (`http://www.microsoft.com/mappoint/products/webservice/default.mspx`), an on-line server (XML Web Service) providing client applications with maps and routing information about outdoor locations. On the other hand, if the location engine is inside the client, then the data store is simply a repository for blocks of location data that can be uploaded by the client at will. This allows the client to pose essentially arbitrary "queries", since the programming language serves as the means of access to the cached location data. Network traffic will be "bursty", but clients are expected to cache whole chunks of the location model, thus eliminating the need to go back to the data store for every query. Predictive prefetching can be used to cache more, reducing latency in answering likely future queries. But, any degree of caching introduces the problem of maintaining consistency when the location data store changes. A client-based location engine can be thought of as a single distributed location engine, which is part resident on every client. HereIAm uses a file repository as the data store, with the location engine compiled into client applications. Alternatively, a system could have a hybrid design, with the location engine split into parts that run on the client system, and other parts that run on the data store server. In a hybrid system, the division of labor between the server and client sides can be viewed as a continuum, and might be tuned during the design process for the data store, or tuned dynamically based on system performance, e.g. delivering larger data chunks when server and network are busier.

Location-Independent Applications: It would be desirable to have a location-based service application that can run against many different location data stores, for example an application to display a building directory on your cell phone or laptop, which will work in any building. Each building is its own administrative jurisdiction, and different buildings may use different systems to represent their location data stores. An interoperability interface to the location data store would be needed to achieve location-independence in the application. If the location engine is in the data store, then this interoperability interface would present generic location queries such as "find nearest", and all the work would have to be done by the server. On the other hand, if the location engine is in the client, then the interoperability interface governs the fetching of chunks of location data, which must all be decodable by the same location engine in the client. A hybrid system would be more challenging to design in this context, with the location engine split into client and server parts, because the interoperability interface would need to dictate the interaction between the parts.

Application-Independent Location Data: Today, most location-based systems are really designed to support a single application. Within a large enterprise, for example, there may be many copies of the building floorplans and campus maps maintained by different branches of the organization for different purposes. For location-based services to become economically viable, it would be desirable to have a single location data store that can serve many applications. This is relatively easy to achieve within an enterprise, in which all data accessors could be expected to use the same client interface software. The structure of the location engine is irrelevant, since all applications use the same software components and can therefore interoperate freely. But for public venues, such as a shopping mall presenting a mall directory to the general public, the expectation must be that different clients may be based on different software systems. This is one reason that publicly available location-based services today generally place the location engine entirely in the server, with a very simple and universal query interface such as a web browser displaying a picture or linking to a specific URL to provide information for the specified location, e.g. the Web-based maps at `http://virtualearth.msn.com`. Unfortunately, the application is therefore designed into the location data store itself, so there is in effect only a single application supported by the data store - while client-system-independence is achieved, application-independence is sacrificed. True application-independence requires a careful design to expose a rich, flexible view of the location data, while not imposing an undue burden of computational intensity on the client.

7 Conclusions

Location-based services are critically dependent on data stores. Today's LBSs are primarily deep but narrow or broad but shallow. As these services grow in breadth and depth, the complex issues of storing and accessing a sometimes messy landscape of location data will present database developers and researchers with interesting challenges.

References

[BMK+00] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven A. Shafer. Easyliving: Technologies for intelligent environments. In *Second International Symposium on Handheld and Ubiquitous Computing (HUC)*, pages 12–29, 2000.

[BS01] Barry Brumitt and Steven A. Shafer. Better living through geometry. *Personal and Ubiquitous Computing*, 5(1):42–45, 2001.

[DdFG01] A. Doucet, N. de Freitas, and N. Gordon. An introduction to sequential monte carlo methods. *Sequential Monte Carlo Methods in Practice*, pages 4–11, 2001.

[DK05] Matt Duckham and Lars Kulik. A formal model of obfuscation and negotiation for location privacy. In *Pervasive*, pages 152–170, 2005.

[HKH05] Ramaswamy Hariharan, John Krumm, and Eric Horvitz. Web-enhanced gps. In *LoCA*, pages 95–104, 2005.

[HT04] Ramaswamy Hariharan and Kentaro Toyama. Project lachesis: Parsing and modeling location histories. In *GIScience*, pages 106–124, 2004.

[KH04a] John Krumm and Ken Hinckley. The nearme wireless proximity server. In *Ubicomp*, pages 283–300, 2004.

[KH04b] John Krumm and Eric Horvitz. LOCADIO: Inferring motion and location from wi-fi signal strengths. In *MobiQuitous*, pages 4–13, 2004.

[KHM+00] John Krumm, Steve Harris, Brian Meyers, Barry Brumitt, Michael Hale, and Steve Shafer. Multi-camera multi-person tracking for easyliving. In *Proceedings of the Third IEEE International Workshop on Visual Surveillance (VS'2000)*, 2000.

[PLG+04] Donald J. Patterson, Lin Liao, Krzysztof Gajos, Michael Collier, Nik Livic, Katherine Olson, Shiaokai Wang, Dieter Fox, and Henry A. Kautz. Opportunity knocks: A system to provide cognitive assistance with transportation services. In *Ubicomp*, pages 433–450, 2004.

ICDE'06

Sponsored by the
IEEE Computer Society

CALL FOR PAPERS

22nd International Conference on Data Engineering

April 3 - April 7, 2006

JW Marriott Buckhead Hotel, Atlanta, Georgia, USA

http://icde06.cc.gatech.edu Mirror: http://icde06.ewi.utwente.nl

Data Engineering deals with the use of engineering techniques and methodologies in the design, development and assessment of information systems for different computing platforms and application environments. The 22nd International Conference on Data Engineering provides a premier forum for sharing and exchanging research and engineering results to problems encountered in today's information society.

We especially encourage submissions that make efforts on (1) Exposing practitioners how the research results and tools can contribution to their everyday problems in practice, and providing them with an early opportunity to evaluate these tools; (2) Raising awareness in the research community of the problems of practical applications of data engineering and promoting the exchange of data engineering technologies and experience among researchers and practitioners; (3) Identifying new issues and directions for future research and development in the data engineering field.

ICDE 2006 invites research submissions on all topics related to data engineering, including but not limited to those listed below:

1. Data Integration, Interoperability, and Metadata
2. Ubiquitous Data Management and Mobile Databases
3. Query processing, query optimization, and data structures
4. Data Privacy and Security
5. Data Mining
6. Semi-structured data and XML databases
7. Distributed, parallel, Peer to Peer databases
8. Web Data Management and Deep Web
9. Scientific and Biological Databases and Bioinformatics
10. Workflow, Web Services
11. Stream processing and sensor databases
12. Data Grids, Data Warehousing, OLAP
13. Temporal, Spatial, and Multimedia databases
14. Database Applications and Experiences
15. Database System Internals and Performance

AWARDS

An award will be given to the best paper. A separate award will be given to the best student paper. Papers eligible for this award must have a (graduate or undergraduate) student listed as the first and contact author, and the majority of the authors must be students. Such submissions must be marked as student papers at the time of submission.

INDUSTRIAL PROGRAM

The conference program will include a number of short papers and invited presentations devoted to industrial developments. Send your papers/proposals electronically, clearly marked as industrial track papers, to the Industrial Program Chair at <chris.bussler[AT]deri[.]org>.

PANELS

Panel proposals must include an abstract, an outline of the panel format, and relevant information about the proposed panelists. Send your proposals electronically by the submission deadline to the Panel Chair at <marek[AT]research[.]telcordia[.]com>.

ADVANCED TECHNOLOGY SEMINARS

Seminar proposals must include an abstract, an outline, a description of the target audience, duration (1.5 or 3 hours), and a short bio of the presenter(s). Send your proposals electronically to the Seminar Chair at <yzhang[AT]csm[.]vu[.]edu[.]au>.

DEMONSTRATIONS

Demonstration proposals should focus on new technology advances in applying databases or new techniques. Proposals must be no more than four double-columned pages, and should give a short description of the demonstrated system, explain what is going to be demonstrated, and state the significance of the contribution to database technology, applications or techniques. Send your proposals electronically to the Demonstration Chairs at
<govi[AT]aztec[.]soft[.]net>, <leo.mark[AT]cc[.]gatech[.]edu>, and <arjen[.]de[.]vries[AT]cwi[.]nl>.

SUBMISSION INFORMATION

Research papers must be prepared in the 8/5"x11" IEEE camera-ready format, with a 12-page limit, and submitted electronically at http://icde06.cc.gatech.edu. All accepted papers will appear in the Proceedings published by the IEEE Computer Society.

IMPORTANT DATES

Abstract Deadline: June 14, 2005
Submission Deadline: June 21, 2005
Notification: September 15, 2005

GENERAL CHAIRS
Ramesh Jain, Univ. California, Irvine, USA
Calton Pu, Georgia Inst. of Technology, USA

PROGRAM CHAIRS
Ling Liu, Georgia Institute of Technology, USA
Andreas Reuter, European Media Lab.Germany
Kyu-Young Whang, KAIST, Korea

LOCAL ARRANGEMENTS
Brian Cooper, Georgia Inst. of Technology,USA
Daniel Rocco, West Georgia University, USA

PUBLICITY CHAIRS
Wei Tang, NCR Corp. USA
Andreas Wombacher, Univ Twente, The Netherlands

TREASURER
E. K. Park, Univ. of Missouri Kansas City, USA

INDUSTRIAL PROGRAM CHAIRS
Christoph Bussler, DERI, Ireland
Fabio Casati, HP, USA

PANEL CHAIR
Marek Rusinkiewicz, Telcordia, USA

SEMINAR CHAIR
Yanchun Zhang, Victoria Univ. Australia

WORKSHOP CHAIR
Roger Barga, Microsoft Research, USA
Xiaofang Zhou, Univ. of Queensland, AUS

DEMONSTRATION CHAIRS
Govi Govindarajan, Aztec, India
Leo Mark, Georgia Inst. of Technology, USA
Arjen de Vries, CWI, The Netherlands

AREA CHAIRS
Tamer Ozsu, Univ. of Waterloo, Canada
Ouri Wolfson, Univ. of Illinois at Chicago, USA
Alfons Kemper, Technische Univ. München, Germany
Elisa Bertino, Purdue University, USA
Johannes Gehrke, Cornell Univ. USA
Yannis Papakonstantinou, UC San Diego, USA
Beng Chin Ooi, National Univ. of Singapore
Paolo Atzeni, Università Roma Tre, Italy
Val Tannen, University of Pennsylvania, USA
Asuman Dogac, METU, Turkey
Donald Kossmann, ETH Zurich, Switzerland
Paul Watson, Univ. of Newcastle, UK
Ming-Syan Chen, National Taiwan Univ.
Masatoshi Yoshikawa, Nagoya Univ. Japan
Sang Kyun Cha, Seoul National Univ. Korea

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398