Querying in highly mobile distributed environments

T. Imielinski and **B. R. Badrinath**
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903
e-mail:{imielins,badri}@cs.rutgers.edu

Abstract

New challenges to the database field brought about by the rapidly growing area of wireless personal communication are presented. Preliminary solutions to two of the major problems, control of update volume and integration of query processing and data acquisition, are discussed along with the simulation results.

1 Introduction

The rapidly expanding technology of cellular communications will give users the capability of accessing information regardless of the location of the user or of the information. It is expected that in the near future, tens of millions of people in the U.S. alone will carry a lightweight, inexpensive terminal that will give them access to a worldwide information network called PCN (Personal Communication Network). These users will be constantly relocating between cells of size much smaller than today (future cell size might be a building or a floor of a building). Thus, within a day, a mobile user may cross the boundaries of these small cells tens and possibly hundreds of times.

Proceedings of the 18th VLDB Conference

Vancouver, British Columbia, Canada 1992

Mobile users will be able to ask ad hoc queries addressed to large databases describing the local area: information about people, places, its geography, services available, etc. Typical queries from mobile users will range from simple requests such as "Where is X?" or "Where is the nearest doctor?" to more complex "Find me the the best route to the hospital with the best traffic conditions." Queries may also request data from a mobile source in a location *transparent* mode: the recent sales figures from a traveling salesman who stores this data in the memory of his mobile palm-top terminal.

Before this vision can be fully realized, a number of new database research problems have to be solved. These problems include the control of massive location updates, the integration of querying and communication, and the partition of knowledge across the mobile network. It is estimated (Meier & Hellstern et al., 1991) that the additional signalling load generated by PCN will be 4-11 times higher for cellular networks than for ISDN and 3-4 times higher for PCN than for cellular networks themselves. Most of this traffic will be due to location updates from constantly relocating users. Additional traffic, which is difficult to predict yet, will be due to the massive data transfers generated by ad-hoc queries.

The goal of this paper is to present a number of new data management problems hopefully of interest to the database community that arise due to the *mobility* of both data sources and data consumers. The management of mobile data is an important part of the emerging area of *mobile or nomadic computing.* It can also be viewed as an integral component of a broader vision of universal and worldwide access to information independent of its form, representation, *location* and *mobility* of users. In addition to the discussion of the new problems, we will also provide some preliminary solutions and simulation results. The full body of results, due to lack of space, will be presented elsewhere.

The work presented in this paper is part of a larger project called DataMan which addresses all of the above questions. The DataMan (a logical successor to WalkMan and WatchMan) will be a distributed knowledgebase capable of handling a large number of queries from hand held terminals[1]. Such a system running on small pocket terminals can replace current pocket organizers by making it possible to access and to process information at *any* location and at *any* time. Information could include maps and navigation, local services, and finally identification of other moving objects as well. The DataMan project will be described elsewhere in more detail.

This paper has two goals:

- Provide preliminary solutions to two of the major problems mentioned above: control of the update volume and the integration of query processing and data acquisition.
- Demonstrate new challenges to the database field brought about by the rapidly growing area of wireless personal communication. Here we present a number of open problems and challenging research questions addressed to the database community.

In Section 2, we first present a general architecture of the cellular system. The next two sections cover in detail the issues of update and query processing. These sections contain the main technical contributions of the paper.

In Section 5, we outline a number of important open questions and future direction of this work. Section 6 summarizes related work and finally, conclusions are presented in Section 7. Preliminary simulation results are described in the appendix.

2 General Architecture

The system consists of a fixed information network extended with a wireless component. The wireless elements include: wireless terminals, base stations and switches. The whole geographic area is partitioned into *cells*. Each cell is covered by a *base station* which is connected to the fixed network and provides a wireless communication link between the mobile user and the rest of the network. The allocated bandwidth of radio spectrum[13] in North America includes frequencies between 870 - 890 kHz (from base station to mobile) and 825-845 kHz (from mobile to base). This bandwidth is subdivided into 666 channels which are shared and reused among base stations which are sufficiently far apart. Currently the average size of a cell is of the order of 1-2 miles in diameter[19]. It is expected, however that with a new generation of wireless systems this size will shrink to possibly 50-200 meters to accommodate a much larger set of users and still benefit from bandwidth sharing and reuse schemes.

Additionally, we will assume that there is a hierarchy L of location servers which are connected among themselves and to the base stations by a standard network (Figure 1). Typically location servers correspond to Mobile Switching Offices and there are about 60-100 base stations "under" a location server. Location servers are responsible for keeping track of addresses of users who are currently residing in the area "below" the location server. These addresses may be stored as exact locations - i.e., the identifier of a cell the user is currently in, or as approximate locations; a zone, or a partition of cells.

Each user (sometimes called mobile terminal) will be permanently registered under one of the location servers; additionally the user may also register as a visitor under some other location server. Base stations will always be aware of mobile terminals which are active within their cells. Location servers, however, do not have to know in which cell a given mobile terminal is currently located - they can always find out by means of paging – a multicast message sent to a subset of base stations "under" the given location server. Since the location server needs to contact the base stations over the fixed network, the cost of broadcasting is equal to the number of base stations to which paging requests are sent[2].

By the users' location we will understand an identifier of a cell in which the user is currently residing. The database storing users' locations will typically be distributed among many location servers. Importantly, the database will almost never store the actual exact location of a user (i.e., the cell id) but either some outdated previous position or a set of "possible locations." Therefore, it will almost always store incompletely specified

[1]The prototype of DataMan will make use of current resources of the Wireless Information Network Laboratory (WINLAB) at Rutgers University which is supported by 18 industrial research sponsors. These resources include a complete cellular communication system with base stations and switching software. We will assume the availability of a small portable low-powered terminal with limited memory (say 8-16MB) which can be switched on and off by software (to minimize power consumption).

[2]The cost of broadcast over the wireless medium does not depend on the number of recipients. However, broadcasting from the location server to the base stations is performed over the fixed network (tree of location servers and base stations). This involves point to point communication from the location server to each of the base stations. Hence, the cost of broadcasting in this case is proportional to the number of base stations.

location data to avoid massive and frequent location updates.

Querying locations will in general lead to combinations of table look-ups and selective broadcasts. For instance, a simple query "Where is John?" asking for a cell number where John currently resides, depending on the addressing protocol may involve accessing John's home location server and then paging him within base stations (corresponding to cells) belonging to the area covered by this location server. If he is not found here then we will have to find the location server under which John currently resides. Several protocols can be used: e.g., broadcasting from the root of the hierarchy of location servers, following the chain of forwarding addresses (if the user leaves them) etc.

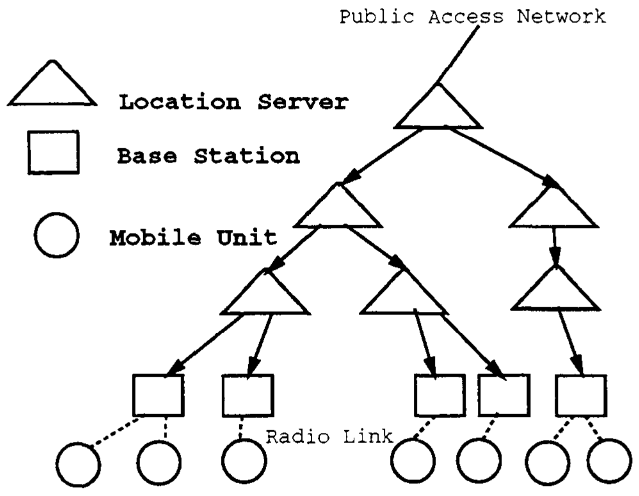


Figure 1: Architecture of Cellular Network

Static and Mobile Databases

Each location server will keep a "home database" of all user related data for users with the permanent residence within the location area administered by the server. Additionally, there will be a visitor database[3] which stores information about users who are not living in the area but are currently visiting the area. Finally, there will be databases related to specialized services such as 1-800-Doctor which would keep information about doctors who are currently residing in some predefined area possibly different from the area covered by the location server. Such specialized databases (views) will be dynamically created as it is the case now for standard telephone services. Databases can also be mobile, especially with the advent of new generations of cellular systems. People will be carrying pocket terminals and running computer programs such as editors, data acquisition packages, etc. while relocating. In such a situation, location servers, specialized servers, etc., will become mobile sources of information, which others in the network may constantly want to access. For instance, the user may be collecting data while moving, and he may not necessarily report all of this new information back to the static server.

In the case when all components of the network become mobile, users and databases have interchangeable roles: each of them can query the other.

2.1 Queries and Updates

The set of users together with their characteristics (including location) can be viewed as a database distributed between different location servers. We will assume a simple object oriented view: our database will have a form of the class of users characterized by a number of methods and instance variables (attributes). One of these methods will determine the location of the user. This class will be stored as a relation which is horizontally partitioned between different location servers according to the "home areas" of users. Further in the paper we will talk extensively about querying and updating such a database. By updates we will really mean *location updates* of the current location of the user. These massive updates will not be transactional in nature and will not require locking. Queries will be standard relational queries which could involve the location attribute both in selection as well as in the target clause.

In the next two sections we address two fundamental questions: how to update and how to query locations in a mobile environment.

3 "Do not tell me - if I do not want to know": Updates

The main idea in this section is that users do not have to inform the location server about each and every change of their location. In other words we should leave a certain degree of ignorance about the state of the system (i.e. user's locations in this case). We postulate that ignorance should be *bounded*, i.e., we should not be too "far off" from the truth. In fact, the *bounded ignorance*[4]

[3]These terms have been introduced earlier in the context of cellular networks[16]

[4]This term is borrowed from [15], where it is used in a different framework — allowing a replicated system to

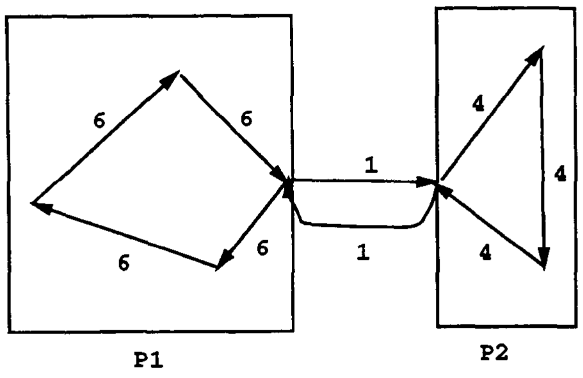


Figure 2: Partitions

approach will guarantee that the real position of the object and the position which is known are always in the same *partition*. Partitions are defined depending on the user mobility patterns and will help to maintain a certain degree of *knowledge* about users' whereabouts (i.e., "I do not know where you are exactly at this time but I know that you are in New Brunswick"). Maintaining such knowledge comes with a cost, but as we will demonstrate it will be much cheaper than keeping the complete information about individual locations.

3.1 Model of agent's mobility within a location server

Intuitively, partitions decrease the overall cost by "gluing together" cells between which the user relocates very often and by separating cells between which the user relocates infrequently. For example, assume that we have gathered the data about the user's mobility between base stations and determined the frequencies with which the user relocates.

In Figure 2, the numbers on the edges indicate the number of times the user relocates between the end locations connected by the edge during a certain interval of time. Edges with a high crossing rate are contained in the same partition and edges with a low crossing rate in separate partitions. In this way the update cost slightly increases (compared to one partition, when the update cost was simply zero) while the query cost significantly decreases, since we only broadcast to base stations within a partition. Obviously there is a tradeoff. Therefore, there is a need for an optimal partitioning which minimizes the expected cost of querying and updating. This task requires knowing more information about users' mobility and patterns of calls.

violate integrity constraints by a bounded amount

Users usually relocate, especially within their home areas according to stable daily patterns (routines). Most of us commute from home to work (and back) along the same route and then have daily routines which are also periodic. The information about each user's mobility is captured by the notion of a *user profile* analogous to credit card and telephone companies storing financial and calling profiles on a per user basis[5].

The user's profile will contain:

1. Probabilities of relocation between any two locations within a location serverbase(l).
2. Average number of calls per unit time
3. Average number of moves per unit time

We assume that the profile is obtained either by monitoring user activity (mobility and frequency of calls) over an extended period of time or/and by getting some of this information directly from the user. In general, of course, probabilities will be approximated by relative frequencies. In the future, we will consider models which allow different patterns of mobility at different time intervals (which is much more realistic, since our daily work routines are different from after hour routines). This model helps us to decide when to notify the location server about the change in our position - it will occur only when we cross the borders of predefined partitions. We assume that partitions will be modified periodically. Shifts in trends of user mobility will be detected and partitions will be modified accordingly at reorganization points. In a way it is analogous to periodic index reorganization.

We will describe several possible algorithms to locate users, assuming some partition based scheme. In the appendix we present preliminary simulation results.

3.2 Algorithms for locating users

In the following discussion we assume that users reside under their home location servers. Thus, the partitions consist of base stations belonging to the same location server. A more general case of user movement across location servers is presented in [4]. We will consider three different strategies within the partition scheme. In each of these strategies, a user informs the location server when he crosses partitions thus incurring update cost, and the search for a user is always within a partition. However, these strategies differ in the way a user is located within the partition. The three strategies are:

[5]Users' profiles will be distributed according to their home location areas. Additionally, we may also store aggregate profiles for specific geographic regions such as a shopping mall, a train station or a football stadium.

1. *Broadcast:* a broadcast is initiated within the partition to locate a user. The cost of locating a user is bounded by the size of the partition, i.e., the number of base stations within the partition.

2. *List:* a list of locations within each partition is maintained. This list is sorted by likelihood of being at a given location. Each location is tried successively. Maximum cost will be incurred when a user is in the least likely location within a partition.

3. *Pointer:* here, each time a user moves to another location, within the same partition, a forwarding address or a pointer is maintained at the previous location. When locating, the pointer is followed from the location first visited in the partition to the current location (similar to call forwarding). If the user visits a location which he has visited before we "cut" the resulting loop. The cost of locating a user is bounded by the the length of this pointer chain; at most the number of base stations within the partition. When a user moves to a different partition, a new pointer chain is maintained in that partition. We assume that leaving forwarding addresses (pointers) at base stations has the same cost as informing a base station of a move. However, we have not included the cost of collapsing pointer loops nor the cost of removing pointers in the previous partition.

In all these strategies, the total cost (update cost + locating cost) depends upon a) the partitions and their sizes, b) user profiles, i.e., the base stations visited, and c) the strategy used.

Simulation results comparing these strategies with respect to the total communication cost (not response time) will be presented in the Appendix.

4 "I do not know but can find out" - Queries

The discussion in the previous section indicates that the database will not usually know the precise location of each user. If the precise location is needed by a query some extra work is needed. This leads to a new paradigm of query processing which involves acquisition of new information in the run time of the query. Before discussing queries and their evaluation we will show how to model the notion of error when dealing with fast changing values.

4.1 Modeling fast changing values - living with errors

Location is an example of a fast changing attribute. Other examples include temperature, speed, pressure etc. in some sensory environment. It is costly to maintain rapidly changing quantities. Therefore we have to be prepared to accept erroneous (outdated) values. In such cases we would like to know, however, what is the margin of error, how to find a better, more precise value and finally, what is the cost of such a precise value finding procedure.

Below we will make a fundamental distinction between the *actual* and *stored* value of a dynamic attribute since most often they will not coincide. The notion of partition will help us to define the margin of error. Various user locating methods (such as paging, pointer and probabilistic addressing schemes) will provide an exact value at additional cost. We will describe all these notions with the help of an example.

Example 1
A user named John will be specified as follows:
Object: John
Partitions: $\{Cell_1, Cell_2\}$, $\{Cell_3, Cell_4, Cell_5\}$
Correctors: Paging with pointer
Predictors: {John is at home, in $Cell_2$ after 6 P.M.}

Additionally (aside from standard attributes such as age, profession, etc), the following methods will be defined on John:
LOC: John.LOC will return the location of John according to the database (which may be outdated and incorrect). For instance, in our case it could be that $John.LOC = Cell_1$
ERR: John.LOC.ERR will return the partition (a set of locations) which is specified by John's profile and contains John.LOC - the stored location of John. If $John.LOC = Cell_1$ then John.LOC.ERR = $\{Cell_1, Cell_2\}$
loc: John.loc returns the actual location of John. This method may involve not only database access but also additional considerable communication in the form of paging and multicasting (so called *corrector methods*).

Correctors will stand for methods which can be used to obtain the exact value of the location, if it is required (that is x.loc). In general, one could have the same correctors associated with all users (this corresponds to one universal protocol of user location) or different correctors associated with classes of users (due to e.g. mobility patterns). For instance, the pointer method may be good for one profile while probabilistic probing superior for another.

Finally, the last element of John's profile illustrates the notion of the "predictor." Location predictor (viewed as a set of rules or a standard procedural method) will compute a *default* location of the user in case the user himself did not notify the location server about *exceptional* behavior. In John's case we stated that at 6 P.M. he is (usually) at home. The location server, at 6 P.M. will try first the user's home to locate him, unless there is a message from the user stating an exception. In such a case, we may have to resort to one of the paging methods.

It turns out that quite often queries can be answered even in the presence of errors therefore incurring no cost resulting from applying correctors. Some other queries may require some of the erroneous data to be corrected. We will discuss this further in the next section.

4.2 Query processing

As we said in the introduction, our queries can range from simple ones such as "where is X" to more complex ones like "Find me a doctor near the campus" or "Find X, Y and Z such that all of them are on the same highway and Y is between X and Z."

These queries differ in the complexity of *location constraints* - i.e., constraints which involve individual location of users. For example, the query "Find me a doctor near the campus" has one non-location based constraint ("doctor") and another which is a unary constraint on location. The query "Find X, Y and Z such that all of them are on the same highway and Y is between X and Z" involves a more complex constraint - a ternary one (between) plus three unary constraints involving individual locations ("on the highway"). Generally, one can get even more complex queries which should recognize "patterns" of moving objects.

The main new problem arising in query processing in the presence of imprecise knowledge about locations of users can be summarized as follows:

How to minimize the communication cost to "find out" the missing information necessary to answer the query?

Without loss of generality we will assume a domain calculus query language. We will concentrate on describing the evaluation strategy for queries written in a "pseudo SQL" form:

SELECT $x_1 \quad ,\ldots,x_m$
FROM Users
WHERE $x_1.loc = l_1 \wedge \ldots \wedge x_n.loc = l_n)$
$\wedge\ C(l_1,\ldots,l_n) \wedge W(x_1,\ldots,x_n)\}$

where $C(l_1,\ldots,l_n)$ is an n-ary constraint imposed on locations $l_1,\ldots,l_n$ and $W(x_1,\ldots,x_n)$ is a constraint on individual objects. Users is a name of the class which stores all instances of users in the system.

The query wants to evaluate a predicate over *actual* locations of users, while the only locations available are $\{x_1.LOC,\ldots,x_m.LOC\}$ with the associated "error" determined by ERR.

A naive strategy to process the query presented above is as follows:

> Find objects $a_1,\ldots,a_n$ such that $W(a_1,\ldots,a_n)$ is satisfied.
> Determine exact locations $a_1.loc,\ldots,a_n.loc$ of each of $a_1,\ldots,a_n$ by paging for each $i(1 <= i <= n)$ the partition $a_i.LOC.ERR$. Check if the result satisfies the constraint $C(l_1,\ldots,l_n)$.

This strategy will, in general, result in too many paging messages. Indeed, paging may be either totally avoided or done only partially. For example, it may turn out that $C(l_1,\ldots,l_n)$ is true *for all* combinations in the cartesian product of $a_1.LOC.ERR,\ldots,a_n.LOC.ERR$. Similarly, the constraint $C(l_1,\ldots,l_n)$ may be false *for all* combinations in the cartesian product of $a_1.LOC.ERR,\ldots,a_n.LOC.ERR$. Finally, we may be able to determine after some partial paging (i.e., to locate a_1 and a_2) that regardless of what the locations of the remaining objects are within corresponding partitions, the constraint is true (or false) saving unnecessary messages. Assuming that the computational costs are much smaller than the communication costs this is a worthwhile gain as the following examples indicate:

> **Example 2** Give the names of doctors (possibly mobile) located near John's current location.
> $\{d$: Near$(d.loc, John.loc) \wedge$ Doctor$(d)\}$ where Near(y,x) is the predicate which is satisfied when cell x is a neighbor of a cell y (either side by side or diagonal).

Let us assume that the database stores L7 as a location of a doctor d and L12 as John's location. Assume also that u is a partition of the doctor's profile which contains L7 and that v is a partition of John's profile which contains L12. In other words we know that the doctor's real location is somewhere within u (d.LOC.ERR = u) and John's real location is somewhere within v (John.LOC.ERR = v).

There are three possible strategies to answer this query: first page both partitions John.LOC.ERR and d.LOC.ERR and then determine whether the final positions of John and

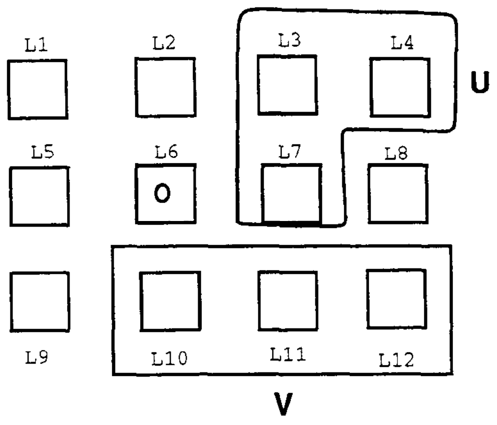


Figure 3: Near By Locations

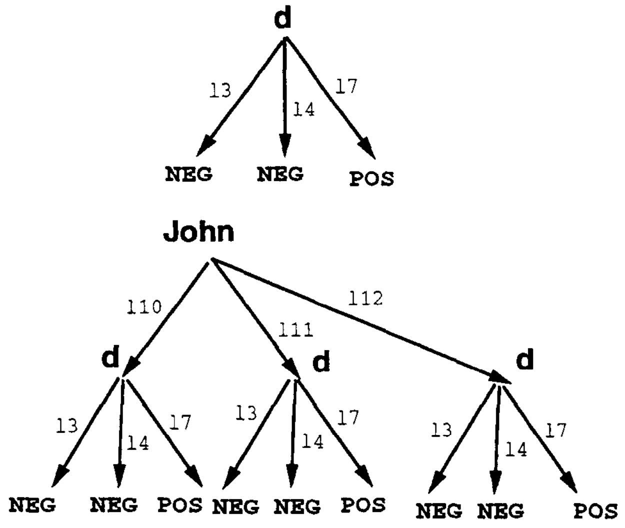


Figure 4: Classification Tree

a doctor are "near by." The other two strategies both start by first determining whether paging is required at all: maybe the predicate Near(x, y) is true or false uniformly for all combinations of locations in John.LOC.ERR and d.LOC.ERR respectively. If this is not the case then we have a choice of first paging John.LOC.ERR or paging d.LOC.ERR. These two strategies may lead to different costs.

Notice that paging u = d.LOC.ERR first is clearly advantageous. No matter what the result of paging is, paging of v = John.LOC.ERR is not necessary. Indeed, if John is in l_7 then no matter where v is, the Near predicate is true. If the doctor is in l_3 or in l_4 then no matter where John is the constraint is false. Therefore, the overall expected cost is 3 in this case. This is represented as the first tree in Figure 4, where the nodes labeled by POS and NEG respectively correspond to situations which satisfy or falsify the query constraint. The first of the trees, which corresponds to paging first the location of a doctor, has only one level since no further paging is necessary.

If we page v first then (without selective paging) we still have in each case to page u. This is why the second tree has two levels. Therefore the expected cost is 6 in this case, shown as the second tree in Figure 4 (if we could page selectively it would be 4, since we could just page l_7 and on the basis of u's presence or absence there conclude the truth value of the constraint). If the Error(x, l_{10}) included l_6 (i.e. the partition for v also included l_6) then we could no longer ignore v after paging u.

In general, we are interested in a (expected) minimal communication cost *strategy* to evaluate the query. By a *strategy* we mean the sequence in which we will page partitions (if we page them at all) $x_k.LOC.ERR$. Such a strategy, in general, can be represented by a tree with nodes corresponding to variables $x_i, i = 1, \ldots, m$, and two special *terminal* nodes called POS (positive) and NEG (negative). POS and NEG respectively will indicate that we can already determine whether the constraint is true or false respectively as indicated in our example. The edges are labeled by conditions $(x_i.loc = c_i)$ where c_i belongs $x_i.LOC.ERR$. In the example, we simply labeled the edges by c_i skipping x_i as clear from the context of the picture.

Each node N of the tree is identified with path(N) which is a conjunction of all conditions along the path from the root to the node. A path $(x_{i_k}.loc = c_{i_k})$ for $k = 1, \ldots, m$ terminates with POS node iff the conjunction of all $(x_{i_k}.loc = c_{i_k})$ along the path implies constraint $C(x_1.loc, \ldots, x_n.loc)$. The same path terminates with NEG if the conjunction of all conditions along the path implies that the constraint $C(x_1.loc, \ldots, x_n.loc)$ is false. For instance, in the first tree of our example, all edges labeled by (d.loc = 13), (d.loc = 14) and (d.loc = 17) are terminal nodes. The first two in NEG, the last in POS. Indeed, d.loc=13 implied logically that the constraint Near(d.loc, John.loc) is false no matter what John.loc is (within John.LOC.ERR).

The cost associated with the node is equal to the number of outgoing edges which do not terminate in NEG node (intuitively this is the cost of paging and we do not have to page the locations which we can determine lead to a failure).

The cost of the path is equal to the sum of the cost of nodes along the path. Each path is equally likely (hence we assume that any combination of locations from $x_1.LOC.ERR$, ..., $x_n.LOC.ERR$

is equally likely).

The expected cost of the path is the sum of costs of all paths weighted by the associated probabilities. The expected cost corresponds to the expected number of paging messages which will have to be sent in order to determine if $x_1.loc, \ldots, x_n.loc$ satisfy the query.

How hard is it to find a tree corresponding to the optimal strategy?

In [3] we demonstrate that the problem is NP-complete, by showing a strong (and somewhat surprising) connection of this problem and the classification problem in machine learning. A greedy heuristic based on the ID3 [21] like algorithm to construct a tree is demonstrated there.

5 New Challenges

Having shown our approaches to the important problems of update and query in mobile environments, we now proceed to a summary of other challenges and open problems which the mobile environment of personal communication systems brings into the database field.

- Incorporating Data Acquisition into Query Processing

  We have provided here a model for incorporating data acquisition into the query answering process.

  Interesting open problems involve the impact of different types of constraints on the selection process for the best classification tree. It should be much easier to find such a tree if the constraint is a conjunction of unary (binary) predicates. Also, there is a need for a more refined cost model which would also take into account the total *time* necessary to broadcast all paging messages.

  Further, we have ignored the possibility that queried objects may reside in the same area and we may be able to locate a number of them within a single broadcast (we have assumed that finding out about each object requires a separate broadcast).

- New Types of Queries

  A highly mobile and distributed environment creates new scenarios for query processing:

  - Queries depending on location of the person asking the query.

    Where is the nearest restaurant? Give me the best directions to the hospital including the current traffic reports. Such queries will involve *geographic* and *real time* data. This will lead us to the question of how to create yellow pages dynamically.
  - Queries depending on direction of movement Knowing our position on a map will not be sufficient; we may need to know the direction of movement to answer a query about the best way to reach the destination.
  - Aggregate Queries

    Such queries could measure global traffic in the area to help in dynamic allocation of resources such as frequency allocation in a particular cell. For example, the area surrounding a football stadium after a game should be assigned more frequencies since the expected activity may be higher there. Aggregate queries will help in establishing such critical zones. Such queries have to take under consideration the road maps in the areas considered especially when detection of high traffic patterns is of interest. This requires careful *integration* with geographic data, street and building maps etc. One may want to detect areas with "no policemen" or with too many taxi cabs (for a taxi dispatcher). This may require complicated matching of patterns of locations of mobile agents in the presence of transient data.

- Querying Transient Data

  Information changes so fast in our application that it may change *during* query evaluation. This creates an interesting research problem on the very basic, semantic level. What is the meaning of the answer to a query? How to compute it and how to augment it? How to incorporate prediction into query processing?

  Frequency is another important resource: for example we cannot issue an arbitrary number of paging actions. Usually there is a limited number of paging channels and if all of them are busy we have to wait. This blocking effect has to be treated as part of the general problem of dynamic resource allocation in the cellular network — some of these problems are currently being addressed in [13].

6 Related Work

Cellular technology is rapidly gaining public acceptance. Wireless computing and access to information in a wireless network will soon follow.

The research work described in this paper is, to our knowledge, one of the few efforts that is addressing issues of information access in a mobile distributed environment.

Current efforts in providing support for wireless computing have focussed on adapting current cellular technology to hand held computers or wireless terminals. Networking software designed for internet use is not applicable to the wireless network problem of dynamically changing addresses; "seamless" handoffs from one network to another require new solutions. Designing network software that is transparent to movement across networks, as well as the design of hardware and user interfaces for hand held computers, is the focus of research at Columbia [8]. Solutions to maintaining a file system for mobile users (where local disks and excessive amounts of RAM are not feasible), and allowing remote paging from file servers in a mobile environment are presented in [11, 12].

Automobile navigation systems also provide limited information about surrounding areas by indicating destinations on a display map. This helps drivers to navigate in unknown areas[5]. Continuous monitoring of moving objects for air-defense coverage was the goal of the SAGE project [18]. Continuous monitoring of moving objects (platforms) by means of active objects [7] has been discussed in [6] Here, the cost of polling for the evaluation of conditions is compared to that of monitoring by means of active objects or triggers.

Cellular networking technology for access to future wireless networks (third generation and beyond) is being investigated at WINLAB in Rutgers[13, 14]. Research is focused on network protocols, cellular packet switching, network arrangements, and spectrum reuse. Further, mobility requires that users and service providers need to be located. The issue of distributed location service for tracking mobile users has first been addressed in [1, 2]. Here, the directory server supports find and move operations. A hierarchical directory structure that minimizes the cost of a sequence of find and move operations is presented. Further, on a move, only nearby directories are updated, so that only approximate information about users' location is maintained. For find operation, nearby users will be able to find the exact location but users far away will have to follow a chain of pointers to find the exact location.

Thus, most of the the current research work has concentrated on cellular networking and architectures for hand held terminals in mobile networks. To our knowledge the research work presented here is the first attempt to address the issues of querying information in wireless and mobile environment (in a much wider context than automobile navigation systems).

7 Conclusions

The vision of universal wireless personal communication offers new challenges to the database field. In this paper we have identified a number of such new research problems arising due to mobility. We provided preliminary solutions to update control and query processing problems ; we have also described a number of additional research issues.

This research effort can be viewed as an integral component of a broader vision: universal and worldwide access to information independent of its form, representation, *location* and *mobility* of users. Work on heterogeneous database systems addresses the first two issues; Our focus is on the other two issues — location and mobility. All these features have to be fully addressed in order to achieve the final goal of *total transparency* of access regardless of form, representation, location and mobility.

Acknowledgments

We are grateful to Irv Traiger for the extensive comments and help in improving the presentation of the paper. We would like to thank David Goodman for letting us know the state of the art in cellular technology and Greg Pollini for explaining the models used in cellular networks. Thanks also go to Phil Bohannon, Asher Mahboob, Miles Murdocca, and Wojtek Szpankowski for various comments on the paper.

References

[1] Baruch Awerbuch and David Peleg, "Concurrent online tracking of mobile users", Proc. ACM SIGCOMM Symposium on Communication, Architectures and Protocols, October 1991.

[2] Baruch Awerbuch and David Peleg,"Sparse Partitions", Proceedings 31'st IEEE Symposium on FOCS, October 1990, St. Louis.

[3] T. Imielinski and B. R. Badrinath, "Querying mobile data," In preparation.

[4] T. Imielinski and B. R. Badrinath, "Adaptive tracking of mobile objects," In preparation.

[5] James L. Buxton, Stanley K. Honey, Wadym E. Suchowerskyj, and Alfred Templehof, "The Travelpilot: A Second Generation Automotive Navigation System," IEEE Transactions on Vehicular Technology, Vol. 40, NO. 1, February 1991.

[6] Sharma Chakravarty and Susan Nesson, "Making an Object-Oriented DBMS Active: Design, Implementation, and Evaluation of a Prototype," International Conference on Extending Database Technology, Venice, Italy, March 1990. (Also in Lecture notes in Computer Science, Vol. 416, pp. 393–406.

[7] Umesh Dayal, et al., "The HiPAC Project:Combining Active databases and timing constraints," Proceedings of SIGMOD Record, March 1988.

[8] Daniel Duchamp, Steven K., Gerald Q. Maguire, "Software technology for wireless mobile computing", IEEE Network Magazine, November 1991.

[9] Garey and Johnson,"A guide to NP complete problems," W. H. Freeman, 1979.

[10] Hyafil, L., and Rivest, R.L., "Constructing optimal binary decision trees is NP-complete", Inform. Process. Lett. 5, No.1, 1976 pp. 15-17.

[11] Bill N. Schilit and Dan Duchamp "Adaptive remote paging for mobile computers" Columbia University, Technical Report, CUCS-004-91, February 1991.

[12] Carl D. Tait and Dan Duchamp, "Service interface and replica management algorithm for mobile file system clients," Proceeding of the Parallel and distributed information systems conference, December 1991.

[13] David J. Goodman, "Trends in Cellular and Cordless Communications" IEEE Communications Magazine June 1991.

[14] David J. Goodman, "Cellular Packet Communications" IEEE Transactions on Communications, VOL. 38. NO 8. August 1990.

[15] N. Krishnakumar and Arthur Bernstein "Bounded Ignorance in distributed systems" In Proc. ACM PODS, Denver, CO, May 1991, pp. 63–74.

[16] Kathleen S. Meier-Hellstern, Eduardo Alonso, "The Use of SS7 and GSM to support high density personal communications," Technical Report WINLAB-TR-24, Rutgers University, December 1991.

[17] Readings in Distributed AI, Edited by Alan Bond and Les Gasser, Morgan Kaufmann, 1988

[18] John F. Jacobs, "SAGE Overview," Annals of the History of Computing, Vol. 5, NO. 4, October 1983.

[19] William Lee "Mobile cellular Telecommunication systems," Mcgraw-Hill, 1989.

[20] C. N. Lo, R. S. Wolff and R. C. Bernhardt, "An estimate of network database transaction volume to support universal personal communication services," Submitted to the 1st International conference on Universal Personal Communications.

[21] J. R. Quinlan "Induction of decision trees", Machine Learning, 1986, pp. 81–106.

[22] John Mylopolous, Alex Borgida, Mathias Jarke, and Manolis Koubarakis, "Telos: Representing knowledge about information systems" ACM TOIS, Vol. 8, No. 4, Oct 90 pp. 325-360.

[23] M. Stonebraker and L. Rowe, "The Design of POSTGRES," Proc. of the ACM SIGMOD conference on Management of Data, Washington, D.C., May 1986.

[24] Gio Wiederhold "Mediators in the architecture for future information systems" Unpublished Manuscript.

[25] Ouri Wolfson and Sushil Jajodia, "Distributed Algorithms for Dynamic replicated data" To appear in ACM PODS, June 1992.

A Simulation studies

Preliminary simulation studies have been conducted to study the effects of using user profiles, in conjunction with the idea of partitioning, on the cost of tracking users by various algorithms described in Section 3.2. The objective of the simulation study is to evaluate various design choices and not an in-depth performance evaluation.

A.1 Simulation model

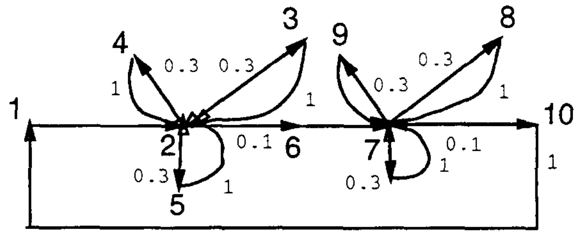
In the simulation model we assume that our queries are simple "calls" to locate a user. Updates are determined by mobility. The ratio of queries to updates, termed the call to mobility ratio, is a critical parameter in our studies.

Further, the important aspect of the performance model is the use of profiles in evaluating various algorithms for locating a user. Since the user profiles vary considerably, it is necessary to consider typical profiles and study how each strategy employed to locate a user performs for these profiles.

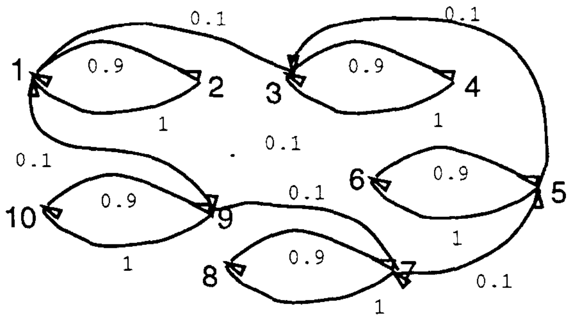
There are a fixed number of base stations among which users move. Moves and calls are generated according to random variables. The mean times between successive calls or successive moves are given by exponentially distributed random variables. The call/mobility ratio is the ratio of the rate (reciprocal of the mean) of calls to the rate of moves. This ratio is an experimental parameter. When a move is generated, the user either decides to stay in the same base station or moves to the next base station which is determined from the user profile. This profile is represented as a transition probability matrix. This matrix gives the probability of a user moving from base station i to j. From the transition probability matrix, it is possible to determine the number of times a user moves from base station i to j within a specified time interval. Thus, within a time interval, it is possible to determine the weights of the edges, i.e., the number of crossings per time interval. Examples of user profiles, with transition probabilities shown as labels on the edges, are shown in Figure 5. For example, in user profile I (partitions are $P_1 = \{1\}$,$P_2 = \{2,3,4,5\}$, $P_3 = \{6\}$, $P_4 = \{7,8,9,10\}$), the user moves randomly within partitions P_2 and P_4. Profile II (partitions are $P_1 = \{1,2\}$,$P_2 = \{3,4\}$, $P_3 = \{5,6\}$, $P_4 = \{7,8\}$, $P_5 = \{9,10\}$), represents the case where a user relocates often between two base stations and then moves to another base station and repeats this behavior.

When a user crosses a partition, the current location is updated, incurring an update cost. The cost of the user informing the base station or location server is more expensive than a base station or location server contacting the user. This is because when a user contacts a base station, there is contention among other users and also the base station needs to authenticate users. Authentication requires messages to be sent between the user and the base station. Hence, more messages and time are needed for a user to contact the base station than the other way around. For purposes of the simulation study, we have chosen the cost of update to be four times that of a call (when the current location is known). This is because of the resource contention due to blocking when communicating from the mobile user to the location server (via the base station). The cost of the call simply reflects the communication cost from the location server to the mobile user. If the location of the user were unknown then the cost would also have to include the cost of search.

Further, a table is maintained in the location server to detect the partition to which a base station belongs. This table gives the partition number for each base station. When a call is generated, the cost of the call, if the current location is not known, will depend upon the strategy employed to locate the user. Thus, for each event, the cost due to either an update or a locate call is accumulated to obtain the total cost incurred by various strategies.



Profile I



Profile II

Figure 5: User Profiles

A.2 Simulation results

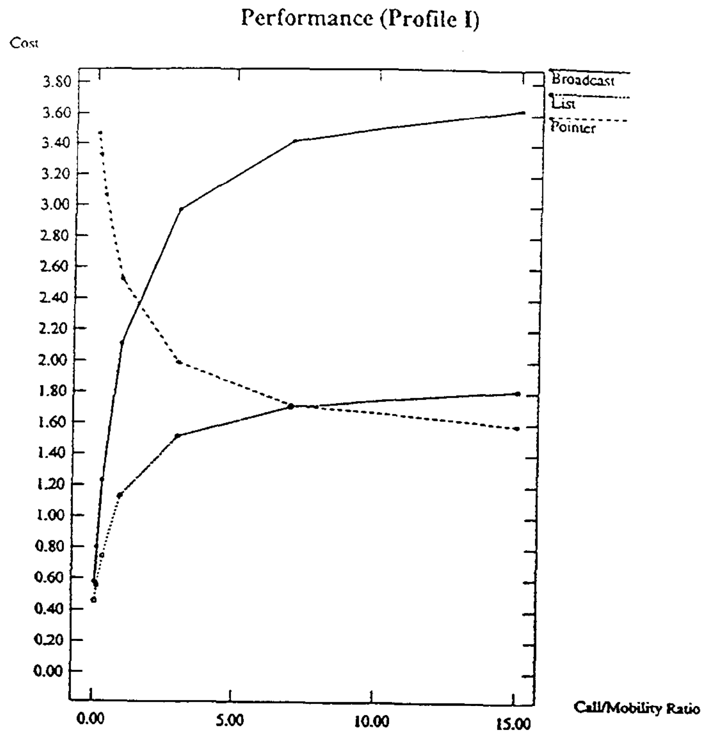
In this section, we present performance results for strategies under the partition scheme. The performance metric employed in comparing various algorithms is the total cost (update cost + location cost) incurred as a function of the call/mobility ratio. The experiment is repeated for different user profiles. The user profiles chosen are shown in Figure 5.

Figure 6 shows the cost per event (calls + moves) (average of 5 runs, for 100000 events) as a function of the call to mobility ratio for various user profiles. Each curve represents the performance of an algorithm. The various algorithms are: Broadcast, List, and Pointer.

Under high call/mobility ratios, the cost of an event for broadcast converges to the number of base stations in the partition; and to the average length of the pointer chain and the average number of base stations in the partition for the pointer based scheme and the list based scheme, respectively. The list based scheme performs better than both the broadcast and the pointer based schemes for Profile I over a wide range of call/mobility ratios. The pointer based scheme performs better than the list based scheme only for a high call/mobility ratio in Profile II. Pointer based scheme incurs a penalty for informing the base stations of the new address for each move. The lower the mobility, the lower the cost of pointers. Thus under high call/mobility ratios, the pointer based scheme incurs a lower cost than under low call/mobility ratios.

The improvement in performance (decrease in cost) as a result of partitioning is very significant compared to that of broadcasting without partitioning. For example, with the above user profiles and the chosen partitioning, the improvement in performance for moderate to high call/mobility ratios with the pointer and list based schemes is two to three fold over schemes that just employ broadcasting. Further, broadcasting without partitioning (i.e., broadcasting to all base stations under a location server) would perform even worse. The pointer based scheme is suited to users exhibiting random movements within each partition and when the call/mobility ratio is moderate to high (Profile I). Under these circumstances, if a list based scheme is used, then a call will be expensive as one has to try all equally likely locations. However, the list based scheme is suited to profiles exhibiting a more deterministic movement within small partitions and when the call/mobility ratio is low to moderate (Profile II). At low call/mobility ratios (high mobility), the list based scheme does not incur any cost due to updates and when a user is often found in one or two most likely locations, the cost of the call is also cheaper. Hence the list based scheme outperforms the pointer and the broadcasting schemes under this condition.

Simulations demonstrate that profile information can drastically reduce the number of location updates. In the future, we plan to perform more extensive simulations and also extend our cost model to include the costs of storing profile information. We also plan to simulate a dynamic updating protocol, where the decision whether to update the location or not is based on the recent history of the call to mobility ratio. This is analogous to dynamic replication schemes [25] where the number of replicated copies of data depends upon the read-write pattern on the data item.



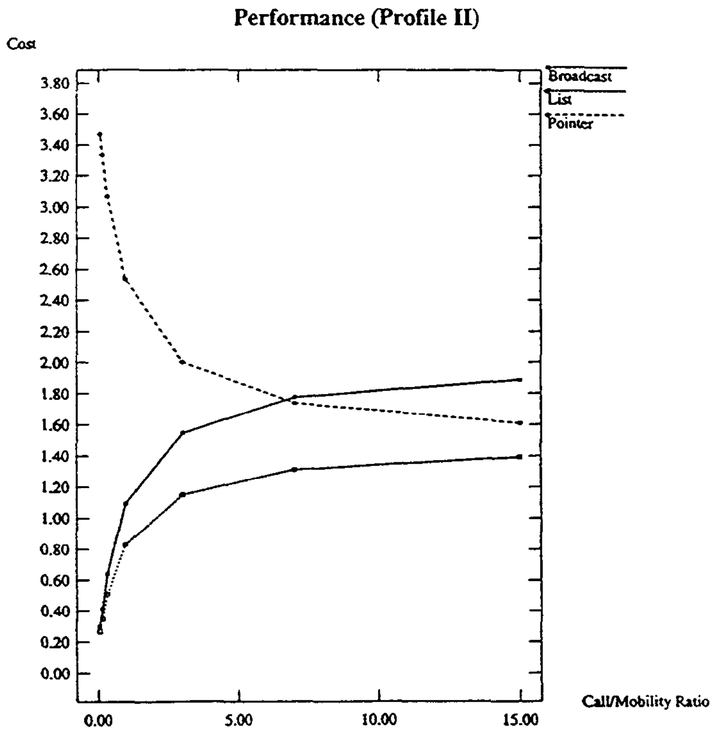


Figure 6: Performance of Various Strategies