Chapter IV
Location-Based Service (LBS) System Analysis and Design

Yuni Xia
Indiana University Purdue University–Indianapolis, USA

Jonathan Munson
IBM T.J. Watson Research Center, USA

David Wood
IBM T.J. Watson Research Center, USA

Alan Cole
IBM T.J. Watson Research Center, USA

ABSTRACT

Advances in GPS, personal locator technology, Internet and network technology, and the rapidly growing number of mobile personal devices are helping to drive the fast growth of mobile e-commerce, we refer to as m-commerce. A special case of m-commerce is enabled with location based services (LBS) where often the actual position of the terminal is used in the service provision. In this chapter, we concentrate on the analysis and design issues and techniques for the LBS. We give a brief introduction of LBS and its applications and present the most important user, system, and infrastructure requirements. We also present the architecture and database design issues in LBS systems and study the performance an LBS system and evaluate its properties.

IINTRODUCTION

Location based services are services that are sensitive to and take advantage of the location of the service user. Any service that makes use of the location of the user can be called a location based service. The location of a person can be determined using a GPS receiver or other technologies, now available in many mobile phone platforms. This position-determination technology (PDT) is generally carried by the person, from which the location must be provided to the location-based service provider. Today, the location-based services are generally hosted in the network, which may pose performance and scalability issues..

The uptake of mobile phones with PDT capabilities continues to grow and most mobile phone users have a phone which can be traced with good accuracy and a lower cost. This new technology has given the LBS market a greater push. Location based services can be divided into four categories:

- Business to business
- Business to consumer
- Consumer to business
- Consumer to consumer

The business to business services include fleet tracking, courier tracking, and others. Business to consumer services include pushed ads based on the location, where a user will receive ads most relevant to the location. Consumer to business services include location based search, where a user is searching for the nearest restaurant, petrol pump, and so forth. A consumer to consumer service is the friend finder service where the user will be alerted if his friend is within a few meters (Jacob, 2007).

Location Based Services Examples

Typical location based services include:

- **Fleet tracking service:** Fleet tracking service can be used to locate and track moving vehicles. This can be very useful for fleet owners/individual vehicle owners. This enables them to pinpoint their vehicles on a map. Taxi fleets can use the location information to calculate the distance traveled, and use that for internal billing. Government can use this service for preventing misuse of government vehicles. Also, the tracker will help in finding the vehicle in case of a theft.
- **Courier tracking:** Courier or costly assets can be tagged with tracking devices so that the position of these can be monitored. In case of freight items, such tracking will allow the user to exactly know where his package is. This avoids manual intervention for data entry and can speed up the handling process. Current freight tracking systems use passive components to track items.
- **Traffic alerts:** Traffic and weather alerts are among the most widely used location services. These alerts provide a traveling person with the latest traffic situation on the road ahead. It also delivers the latest weather updates for the subscriber's current region. Traffic radios are very popular in many countries, but location based traffic alerts can provide customer specific traffic alerts only when he needs an alert.
- **Location sensitive advertising:** LBS can be used to provide location sensitive advertising. Consider a scenario where a traveler gets a message about the nearest hotels as soon as he lands in an airport. In a shopping mall, the service can be used to push information about shops offering a discount. Everyone within a fixed distance from the mall will get the promotional message. This can increase sales and brand awareness.
- **Emergency assistance:** Emergency assistance is the most useful and life saving among all location based services. In case of an accident, a panic button can be pressed, which will send out the location information where the accident took place to nearest control center/ambulance/patrol car. This will help the police/paramedics to reach the accident scene earlier and save lives.
- **E911 services:** Location based services got a boost when the E-911 enhancements were suggested to the 911 system in the USA. E-911 specifies that all operators must provide the location of a 911 caller so that the emergency team can reach the spot faster. This brought about many technological and business changes. Mobile locating technology enables and improves E911 services with increased accuracy and faster query time.
- **Location based yellow page look up:** When you are traveling you will require many kinds

of information, like the nearest restaurant, medical shop, ATM counter, hospital, tourist places, and so forth. Such information can be provided location specific, if we know the traveler's location. The user can request for such services using his mobile and he will get the result in the mobile itself, either using SMS or GPRS.

- **Friend finder:** Friend finding services allows the user to keep track of his friends' locations. A user can query to find out where his friends are right now. This service has some implications on privacy, but still this is an interesting service. Alerts can be provided to users when his or her friends come within a specific distance.
- **Real-time dynamic routing and navigation service:** Electronic maps greatly benefit from knowing the location of a user. The information can be used to plot the location of the user. The map also can show real time direction tips to reach a destination. With a voice enabled LBS-navigation system, the device will give you directions, asking you to "turn right" or "turn left" when a deviation comes.
- **Child/pet tracking:** Tracking devices can be installed in collars or lockets and can be used to track children and pets. Software can be made in such a way that the users can set boundaries. When the child or the pet approaches the boundary or crosses it, the software can raise an alarm or can send an alert message informing the parent or owner.
- Location based billing: Location based billing can be used for promotional activities. The concept is that the billing tariff depends in part on the location of the user. Operators can club with other service providers to make maximum use of location based billing.
- **Games:** Location based games can be very attractive, especially to youth. Treasure hunts, urban combat games, and find me are some famous location based games (Jacob, 2007).

LBS System Components

LBS is an intersection of three technologies:

- New information and communication technologies (NICTS) such as the mobile telecommunication system and hand held devices;
- Internet; and
- Geographic information systems (GIS) with spatial databases.

LBS give the possibility of a two way communication and interaction. Therefore, the user tells the service provider his actual context, like the kind of information he needs, his preferences, and his position. This helps the provider of such location services to deliver information tailored to the user needs (Steiniger, Neun, & Edwardes, 2007).

If the user wants to use a location based service, different infrastructure elements are necessary. The basic components in LBS are:

- **Mobile devices:** A tool for the user to request the needed information. The results can be given by speech, using pictures, text, and so on. Possible devices are PDA's, mobile phones, laptops, and so on, but the device can also be a navigation unit of car or a toll box for road pricing in a truck.
- **Communication network:** The second component is the mobile network, which transfers the user data and service request from the mobile terminal to the service provider and then the requested information back to the user.
- **Positioning determination technology (PDT) component:** For the processing of a service, the user position usually has to be determined. The user position can be obtained either by using the mobile communication network or by using the global positioning system (GPS). Further possibilities to determine the position are WLAN stations, active badges, or radio beacons. The latter position-

ing methods can be especially used for indoor navigation, like in a museum. If the position is not determined automatically it can be also specified manually by the user.

- **Service and application provider:** The service provider offers a number of different services to the user and is responsible for the service request processing. Such services offer the calculation of the position, finding a route, searching yellow pages with respect to position, or searching specific information on objects of user interest (e.g., a bird in wild life park) and so forth.
- **Data and content provider:** Service providers will usually not store and maintain all the information which can be requested by users. Therefore, geographic base data and location information data will usually be requested from the maintaining authority (e.g., mapping agencies) or business and industry partners (e.g., yellow pages, traffic companies) (Steiniger, Neun, & Edwardes, 2007).

LOCATION BASED SERVICE REQUIREMENTS ANALYSIS

Besides the general requirements for m-commerce, location-based services are subject to a set of specific requirements. The requirements can be classified into the following categories: user (functional), usability, reliability, privacy, location infrastructure, and interoperability. These requirements cover some basic issues in location-based services (Tsalgatidou, Veijalainen, Markkula, Katasonov, & Hadjiefthymiades, 2003).

User Requirements: Functional Requirements

User requirements are formed on mobile user exceptions of a LBS. These requirements are the source for the functional requirements of an LBS application. Hermann and Heidmann analyzed a specific LBS application, namely location based fair guide. They conducted interviews with a set of potential users of such a system and a set of visitors of a book fair in order to investigate the user requirements empirically (Hermann & Heidmann, 2002). The list of features the interviewees considered indispensable includes browsing of spatial information (locations of facilities and exhibitors) connected with catalogue data, activity planning, and way finding. The same requirements apply, of course, to any LBS serving a visitor in an unfamiliar environment, whether it is a fair, museum, city, or country. Therefore, the main LBS functional requirements are the following:

- Browsing of spatial information, for example, in the form of a city map.
- **Navigation:** The user must be able to acquire directions and guidance for reaching a specific point of interest.
- Access to catalogue data, for example, names of restaurants and their description like menu, range of prices, opening hours, and so forth. Catalogue data must be spatially referenced so the facilities in the catalogue can be represented on the map and directions for reaching them can be acquired.
- **Location-based access:** The user must be able to access map and catalogue data based on his/her present location. If locating the user cannot be resolved automatically by the system, the user must be provided with the option of manually entering his/her location. The system should be able to handle such requests through geocoding procedures.
- **Personalized access:** The user must be able to specify the type of information s/he needs, for example, a map depicting just the street network or a map including various points of interest. A profile mechanism should also be provided so that the user does not have to input his preferences every time s/he uses the service.

- **Fast access:** The user must be able to run various queries, for example, for nearby restaurants or even for the nearby vegetarian restaurants that are opened after 9 p.m. As can be seen, the LBS user interface must deliver to the user various information including a static map image of the desired area, the location of the user, and possibly, locations of other mobile objects, descriptive information on points of interest, route information, and so forth. Clearly, all this information cannot be presented at once, so an implied requirement is that LBS must provide a set of alternative views:
- **Geographical view:** Graphical representation of a geographical area, usually in the form of a map.
- **View corresponding visual perceptions:** The view as the user could see it if s/he went to the place of interest. This view can be implemented with photographs or 3D modeling.
- **Geographical information view:** Textual description of points of interest enabling the user to browse through it.
- **Status view:** Description of the current state of the user in the form of picture, text of voice.
- **Context view:** Tied to information representation about close objects and the user possibilities in the current context.
- **Route view:** Graphical representation of a route from one point to another.
- **Logistic view:** Abstract route model, showing only intermediate and final points, but no specific distances and directions
- **Guidance view:** Turn-by-turn instruction for the user, usually in a text form and also possibly as voice.

Usability Requirements

Mobile computing environment has certain features that impose restrictions. The properties of mobile networks are: (relatively) low bandwidth, strong bandwidth variability, long latency, unpredictable disconnections, and communication autonomy. The properties of mobile terminals are: small and low-resolution displays, limited input capabilities, limited computing power, limited power, and small memory size. The practical conditions, when and where the mobile devices are used, bring also additional restrictions. The using conditions cannot be expected to be constant, as usually is the case in "desktop" conditions. The mobile users are typically in very unstable environments, in varying conditions, where their cognitive capacity is demanded for other tasks as well. All these restrictions have to be taken very carefully into account when designing LBS. Some of the implied requirements are:

- Not very intensive use of mobile network and minimal volume of transmitted data;
- Possibility to offline operation; and
- User interface should be very simple and user friendly and the amount of presented information content limited and well specified.

Reliability Requirements

LBS are intended mainly for traveling people as a tool providing support in making decisions about where to go. Therefore, wrong information may mean wrong decisions, lost time and, as a result, client anger in the best case and a court examination in the worse case. Consider, for example, a case where LBS misled the user when he needed an urgent medical assistance. Reliability requirements can be divided into the following sub-requirements:

- **Data reliability:** The most important since LBS are built around spatial and catalogue data and incorrect or missed data can easily lead to problems.
- **Software reliability:** Applies to both server and client sides.
- **Appropriateness and precision of exploited algorithms and methods:** Depending on the task in hand, for example, user positioning precision may be sufficient or insufficient, calculating distances by straight line may be sufficient for some tasks but insufficient for

others (the road network thus must be taken into account).

Privacy Requirements

Privacy handling is a major issue in LBS deployment and provision and a critical success factor to the wide acceptance of this technology framework. The term privacy handling consolidates issues like ownership of location information, use of location information, disclosure to service providers, and so forth. Skepticism arises as to where and how privacy handling should take place within the LBS provision chain.

Existing proposals from operators and standardization bodies (e.g., OMA, 3GPP) specify a priority scheme whereby the core network elements (e.g., home location registers) have master control on location information. The provision/disclosure of such information to other entities (e.g., location servers, LBS serving nodes, ASPs) is subject to subscriber needs (e.g., registration information) and regulatory frameworks.

Location Infrastructure Requirements

Location-based service consists of roughly two phases, determining the position of the customer and providing service or contents based on the position. For the location method at least the following requirements can be listed:

- The method should provide good accuracy subject to the requirements of the application and the respective cost.
- The area where the mobile device can be located should be as large as possible; it should be possible to determine in advance where the device can be located; ideally, this should happen within the whole coverage area of mobile networks.
- The method should be fast: also in this respect, the applications have different demands, for example, the emergency applications and car navigation systems have higher demands than "FIND restaurant"—type applications.
- The location method should not generate too much signaling load within the mobile network.
- The effects of adding location method support to a terminal and using it should be minimal, that is, it should not increase its size, weight, power consumption, or price.
- The location method should have a minimum impact on the mobile network in terms of complexity and cost.
- It should be possible to locate all mobile devices irrespective of their type and whether they are roaming or not.
- It should be possible to locate a large group of mobile devices at the same time.
- Consumer privacy must be ensured, by, for example, providing means for the consumer to turn off the locating feature of the terminal.

Service Interoperability Requirements

The interoperability should be assured on all levels of the system architecture. The LBS platform should be interoperable with several types of terminals (e.g., PDAs, GSM terminals) and positioning infrastructures (e.g., indoor, GPS). The platform should be able to handle different coordinate reference systems (e.g., WGS-84 and local systems) in order be able to utilize geographic data available in existing GIS databases. The seamless service provision when different infrastructures are visited by the user is also a requirement in the highly diversified modern telecomm landscape (Tsalgatidou et al., 2003).

LOCATION BASED SERVICE SYSTEM DESIGN

The work flow of LBS is shown as in Figure 1. Where mobile users need information service or

the management center needs to track them, the embedded positioning device, that is, GPS provides current positioning information and transmits it to the center. The service requirements are analyzed by the GPS central GIS server to get results, which is transmitted to the mobile users. LBS generally include two processes. One is to collect the position; the other is to provide service according to user requirement (Hightower & Borriello, 2001; Chen, Zhang, Sun, & Luo, 2004)

LBS System Architecture Design

In this section, we will focus on the architecture design of location utility, which is a typical distributed LBS system developed for supporting a large class of applications that involve responding to the movements of large numbers of mobile users. Other architectures have been proposed in the literature (Agre, Akinyemi, Ji, Masuoka, & Thakkar, 2002); we choose to present the location utility architecture for the simplicity and efficiency. Location utility was proposed and developed by the IBM Watson Research Center. As shown in Figure 2, it consists of a control center (LUCC) and a set of local servers (LULS). This distributed architecture is proposed for the purpose of scalablity. Application rules/functions are distributed to all local server nodes, but each local server handles the subset of the users associated with the local node.

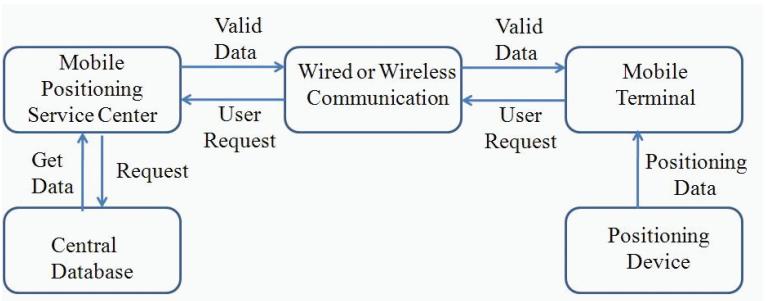
Let us take a simple example and see how this architecture works. Suppose there is a location based service that monitors a few restricted areas and sends alarms when an unauthorized person enters them. This service is first subscribed through the control center. The control center then identifies every local server whose cover region intersects with the restricted areas and the service is subscribed in these relevant local servers as well. The rule subscribed in each local server is:

If (!IsMemberOf(AuthorizedPersonnel)) and Position.ContainedInPolygonClass(RestrictedArea) sendAlarm()

This rule is evaluated at the local servers. When it is true, an alarm is sent to the control center and the control center will send an alarm to the application.

Local server nodes receive the raw subscriber position information from the positioning technology and evaluate the rules assigned to them. Rules that evaluate to true generate reports, which are sent to the application that subscribed to the rule. If the application subscribed to a rule through a local server directly, the report is forward to the application directly; if the application subscribed through the control center, the report is forwarded to the control center, which will in turn forward it to the application. To be able to evaluate rules that involve multiple subscribers, local server nodes may themselves subscribe to rules in other local servers. Where possible, local server nodes are deployed in alignment with the network infrastructure in order

Figure 1. LBS work flow



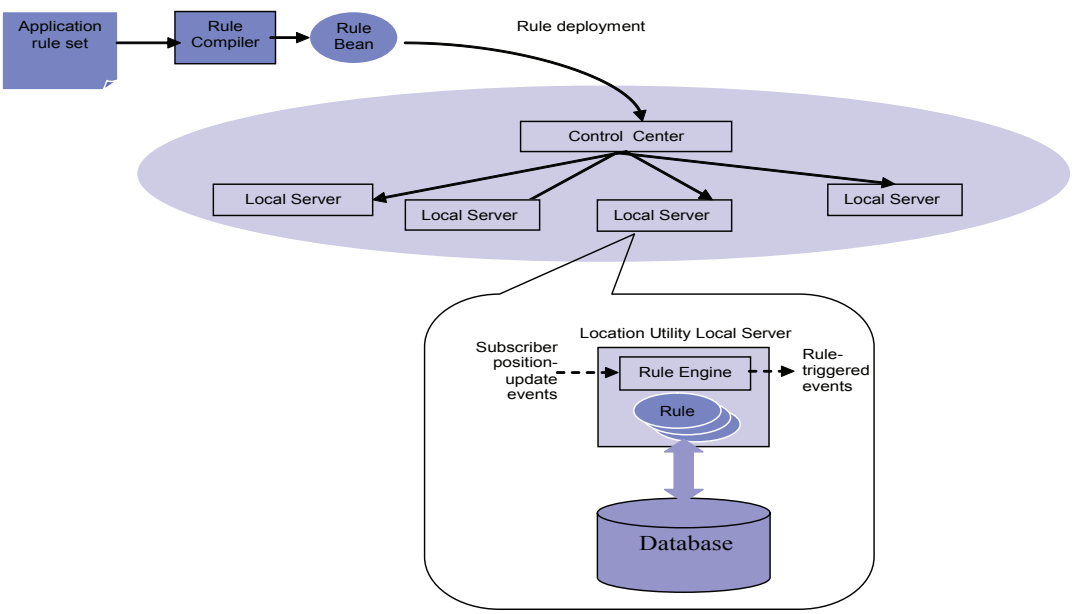
to take advantage of local information about the user presence.

A local server may not evaluate all rules distributed to it. Knowing which users it is currently serving and the geographic region it serves, a local server can, for some rules, decide that the rules can never evaluate to true, and "deactivate" them to reduce its own load. In short, rules that refer to geometries that lie outside a local server's service region and rules that refer to no specific subscribers currently not served by a local server, will not be evaluated by that local server. Because the set of subscribers served by a local server is dynamic, a rule's evaluation status is dynamic. A rule pertaining to a particular subscriber is activated in a local server node when the subscriber moves into the region covered by the node. Any subscriber resources attached to the subscriber are also moved to that local server.

The control center is the locus of control and management functions, and is the primary point of contact for applications to subscribe to rules and set up application resources. It also provides an interface for retrieving subscriber positions and any resources applications may have created for them. When an application subscribes to a rule through the control center, the control center records the subscription information and forwards the rule to all local server nodes. Reports from these rules are sent from the local server nodes to the control center, which aggregates them and forwards them to the application. Application resources set up through the control center are also forwarded to the local server nodes, but resources that have associated geometries are forwarded only to those local server nodes whose geographical domain intersects them. The control center maintains a table of control center information to enable this.

The application interface to the control center includes a function to immediately retrieve rule evaluation results, which will be retrieved from the local server nodes where they are maintained. For those rules that are evaluated at all local server nodes, all nodes will be requested to return the results. For those rules that are evaluated only at certain nodes, because they apply only to the domains of

Figure 2. Location-based service system architecture



certain nodes, only those nodes will be requested to return the results.

When a subscriber moves from the region served by one local server to that of another, the resources that applications may have created for the subscriber must be transferred from one to the other. The control center must also be notified of the new serving local server. A detailed description of the system architecture and components can be found in Munson, Cole, Duri, and Wood (2003).

LBS Database Design

In this section, we present a data model for representing moving objects in database systems. It is called the moving objects spatio-temporal (MOST) data model (Sistla, Wolfson, Chamberlain, & Dao, 1997).

Existing database management systems (DBMSs) are not well equipped to handle continuously changing data, such as the position of moving objects. The reason for this is that in databases, data is assumed to be constant unless it is explicitly modified. For example, if the salary field is 30K, then this salary is assumed to hold (i.e., 30K is returned in response to queries) until explicitly updated. Thus, in order to represent moving objects (e.g., cars) in a database, and answer queries about their position (e.g., How far is the car with license plate RWW860 from the nearest hospital?), the car's position has to be continuously updated. This is unsatisfactory since either the position is updated very frequently (which would impose a serious performance and wireless-bandwidth overhead), or, the answer to queries is outdated. Furthermore, it is possible that due to disconnection, an object cannot continuously update its position. One common solution is to represent the position as a function of time; it changes as time passes, even without an explicit update (Saltenis, Jensen, Leutenegger, & Lopez, 2000; Tao, Papadias, & Sun, 2003). For example, the position of a car is given as a function of its motion vector (e.g., north, at 60 miles/hour). In other words, a higher level of data abstraction is considered, where an object's motion vector (rather than its position) is represented as an attribute of the object. Obviously, the motion vector of an object can change (thus it can be updated), but in most cases it does so less frequently than the position of the object.

A data model called moving objects spatio-temporal was proposed for databases with dynamic attributes, that is, attributes that change continuously as a function of time, without being explicitly updated. In other words, the answer to a query depends not only on the database contents, but also on the time at which the query is entered. Furthermore, we explain how to incorporate dynamic attributes in existing data models and what capabilities need to be added to existing query processing systems to deal with dynamic attributes. Clearly, the MOST model enables queries that refer to future values of dynamic attributes, namely future queries. For example, consider an air-traffic control application, and suppose that each object in the database represents an aircraft and its position. Then the query Q = "retrieve all the airplanes that will come within 30 miles of the airport in the next 10 minutes" can be answered in the MOST model.

Continuous queries are another topic that requires new consideration in our model. For example, suppose that there is a table called MOTELS giving for each motel its geographic-coordinates, room-price, and availability. Consider a moving car issuing a query such as "Display motels (with availability and cost) within a radius of 5 miles," and suppose that the query is continuous, that is, the car requests the answer to the query to be continuously updated. Observe that the answer changes with the car movement. When and how often should the query be reevaluated? Our query processing algorithm facilitates a single evaluation of the query; reevaluation has to occur only if the motion vector of the car changes.

A. Data Model

The traditional database model is as follows. A *database* is a set of object-classes. A special database

object called *time* gives the current time at every instant; its domain is the set of natural numbers, and its value increases by one in each clock tick. An *object-class* is a set of attributes. For example, MOTELS is an object class with attributes name, location, number-of-rooms, price-per room, and so forth.

Some object-classes are designated as *spatial*. A spatial object class has three attributes called X.POSITION, Y.POSITION, and Z.POSITION, denoting the object's position in space. The spatial object classes have a set of spatial methods associated with them. Each such method takes spatial objects as arguments. Intuitively, these methods represent spatial relationships among the objects at a certain point in time, and they return true or false, indicating whether or not the relationship is satisfied at the time. For example, INSIDE(o,P) and OUTSIDE(o,P) are spatial relations. Each one of them takes as arguments a point-object o and a polygon-object P in a database state; and it indicates whether or not o is inside (outside) the polygon P in that state. Another example of a spatial relation is WITHIN-ASPHERE (r, o1, o2, … on). Its first argument is a real number, and its remaining arguments are point-objects in the database. WITHIN-ASPHERE indicates whether or not the point-objects can be enclosed within a sphere of radius r. There may also be methods that return an integer value. For example, the method DIST (o1, o2) takes as arguments two point-objects and returns the distance between the point-objects.

To model moving objects, the notion of a dynamic attribute was introduced.

B. Dynamic Attributes

Each attribute of an object-class is either static or dynamic. Intuitively, a static attribute of an object is an attribute in the traditional sense, that is, it changes only when an explicit update of the database occurs; in contrast, a dynamic attribute changes over time according to some given function, even if it is not explicitly updated. For example, consider a moving object whose position in a two dimensional space at any point in time is given by values of the x, y coordinates. Then each one of the object's coordinates is a dynamic attribute.

Formally, a *dynamic attribute* A is represented by three sub-attributes, A.value, A.updatetime, and A.function, where A.function is a function of a single variable that has value 0 at time 0. The *value* of a dynamic attribute depends on the time, and it is defined as follows. At time A.updatetime the value of A is A.value, and until the next update of A the value of A. At time A.time+t0 is given by A.value + A.function (t0). An explicit update of a dynamic attribute may change its value sub-attribute, or its function sub-attribute, or both sub-attributes.

In addition to querying the value of a dynamic attribute, a user can query each sub-attribute independently. Thus, the user can ask for the objects for which X.POSITION.function=5*t, that is, the objects whose speed in the x direction is 5.

There are two possible interpretations of A.updatetime, corresponding to valid-time and transaction-time. In the first interpretation, it is the time at which the update occurred in the real world system being modeled, for example, the time at which the vehicle changed its motion vector. In this case, along with the update, the sensor has to send to the database A.updatetime. In the second interpretation, A.updatetime is simply the time-stamp when the update was committed by the DBMS. In this chapter, it is assumed that the database is updated instantaneously, that is, the valid-time and transaction-time are equal.

When a dynamic attribute is queried, the answer returned by the DBMS consists of the value of the attribute at the time the query is entered. In this sense, our model is different than existing database systems, since, unless an attribute has been explicitly updated, a DBMS returns the same value for the attribute, independently of the time at which the query is posed. So, for example, in our model the answer to the query: "retrieve the cur-

rent X position of object O" depends on the value of the dynamic attribute X.POSITION at the time at which the query is posed. In other words, the answer may be different for time-points t1 and t2, even though the database has not been explicitly updated between these two time-points (Sistla, Wolfson, Chamberlain, & Dao, 1997).

SYSTEM PERFORMANCE ANALYSIS AND PLANNING: A CASE STUDY

In this section, we present the results of some experiments to analyze the performance of the local server in location utility LBS systems. The execution time needed to process the most common requests or operations in local based services are measured and the most time consuming computations and operations are identified. The data obtained in these experiments will not only identify time consuming operations or potential bottlenecks so to optimize the system, but also provide reference for hardware selection, performance and capacity planning, and tailoring the various components to suit the complex needs of the application and the system.

This section also offers a quantitative way of finding the optimal cell size for deploying local servers for the local based services system. The model relates several factors such as local server processing capacity, control center capacity, mobile users' arrival and departure rate, and the traffic types. The goal of this work is to specify an approach of computing optimal coverage region for the local server that satisfies two criteria: (i) queries/services can be finished in timely way and (ii) the number of crossover or handoffs should not become overwhelming. It gives a set of basic results and offers an understanding of cell size tradeoffs in location-based service systems. It also provides a general guideline for choosing the covering range when deploying local based services local servers.

Performance Analysis Environment Setup

In all the experiments, a 2.4Ghz Pentium III machine with 1GB of memory is used. Due to the unavailability of actual object movement data, a synthetic dataset generated by the City Simulator is used. The City Simulator was developed by IBM Almaden Research Center to create dynamic spatial data simulating the motion of up to 1 million people. The data format generated by the City Simulator is as follows:

```
1, 0.037288, 530.8,164.0,18.0      cycle = 1
2, 0.074932, 299.4,143.6,24.0      cycle = 1
3, 0.096154, 312.6,805.0,5.3       cycle = 1
...
1, 29.926125, 529.2,164.0,18.0     cycle = 2
2, 29.957042, 299.4,143.6,24.0     cycle = 2
3, 29.978109, 312.2,804.4,5.3      cycle = 2
   ...
```

Each record contains a unique person ID, a time stamp, and the x,y,z coordinates. These data are stored in spatial position record (SPR) files. In the experiments, the LULS reads the SPRs of each object from SPR files, simulating the locations of users and the arrival of these events at the LULS. It is assumed that LULS has the latest locations of each object. In LULS, location services are expected to be implemented through rules or rule combinations. Take the example of the location-based advertising/promotion service, when a preferred customer enters the proximity of certain stores, coupons or advertisements will be sent to him/her. The query of finding such customers in LULS would be a rule combination as following:

IsMemberOf (PreferredCustomerClass) and ContainedInPolygonClass(StoreClass).

For the underlying database system, the IBM's DB2 relational database system with spatial extender is used. Resources such as polygons and points,

which represent places of interest such as restaurants, hotels, gas stations, and other static locations of interest, are stored in the spatial database.

The notion of RPS (rule per second) was proposed to be the benchmark parameter that should be measured. This benchmark is similar to MIPS (million instructions per second) performance parameter for CPU and TPS (transactions per second) for database systems. To get RPS for different rules, the time needed to process each rule is measured.

Functions Measured

The following are the most common rules/operations for location-based services. All the rules here serve as the basic predicates or operations in the system. Complicated applications and services can be built based on them.

- **HasIdentity:** Test if the subscriber has the given identity ID.
- **HasValue:** Test if there is a variable with the give name. Both cases—when the variable exists and when it does not exist, are tested.
- **IsMemberOf(subscriberClass):** Test if the subscriber is a member of one or more of the given classes. To test this rule, a MembershipGenerator was developed for generating membership with specified size. Four memberships were generated, with size 100, 1K, 10K, and 100K respectively.
- **IsTrue/IsFalse:** IsTrue/IsFalse is the simplest rule. By measuring the time needed for processing these rules, we get an idea of the pure overhead of going through the rule evaluating process.
- **Logical rules:** Performs basic logic operations including AND, OR, NOT.
- **Relational:** Performs basic comparison operations including EQ (=), GE (>=), GT (>), LE (<=), LT (<), and NE (!=).
- DistanceFrom
  1. **DistanceFrom.Point:** Returns the great circle distance from the given subscriber position to the specified point.
  2. **DistanceFrom.Subscriber:** Returns the great circle distance from the given subscriber position to the specified subscriber.
- **ContainedIn.Polygon:** Test if the subscriber position is contained in the given polygon. The polygon is a resource that stored in database.
- **ContainedIn.PolygonClass:** Test if the subscriber position is contained in the given set of polygons, which are referred to via a class identifier. To test ContainedIn.PolygonClass, a PolygonGenerator was developed for generating a number of uniformly distributed polygons with specified area size. These polygons can then be loaded into the database as one class.
- **WithinDistanceOf.Point:** Test if the subscriber position is within the given distance from the given point.
- **WithinDistanceOf.PointClass:** Test if the subscriber position is within the given distance from the given set of points, which are referred to via a class identifier. To test WithinDistanceOf.PointClass, a PointGenerator was developed for generating a number of uniformly distributed points within the specified space. These points are then loaded into the database as one class.
- **WithinDistanceOf.Subscriber:** Test if the subscriber position is within the given distance from the position of the given subscriber.
- **WithinDistanceOf.SubscriberClass:** Test if the subscriber position is within the given distance from the position of one or more of the given subscribers, referred to via a class identifier.

Execution Time

The process of measuring the execution_time is as following: the rules to be tested are subscribed into LULS, and then SPRS are fed in through a socket. The time needed to evaluate each rule is then measured.

The execution time for each rule/operation is shown in Table 1. As we can see from the table, the processing time of these rules varies significantly, ranging from 0.26ms to almost 300ms.

Generally speaking, there are two types of rules: rules that need access to the database, for example, ContainedIn.PolygonClass and WithinDistance. PointClass, and rules that do not need access to the database, such as HasIdentity. As shown in Table 1, rules that access the database usually take much more time than rules that do not. That is easy to understand since database operations tend to be I/O intensive and time consuming.

Performance Improvements

A. Use Spatial Index

Since rules that access the database take a much longer time than rules that do not, this indicates that database access is a costly operation. In order to improve the database performance, spatial indexes are used. DB2 spatial extender provides a grid index, which was used for measuring the process time of two rules: ContainedIn.PolygonClass and WithinDistanceOf.PointClass. It showed that when the database has only a small number of geometries, like less than 1000 polygons or points, the index does not improve the performance or may even harm it. The reason is that with an index, the database may have to go through the index to reach the actual data, and the index access causes extra I/Os. However, as the number of geometries in the database gets larger, the index starts to show its benefit. The index can efficiently help with pruning and reduce the number of objects queries that need access. Therefore, the more geometries in the database, the more significant the index improvement is. Figure 3 shows that for 100,000 polygons, the processing time of ContainIn.PolygonClass without index is between 270 to 300ms, while with index; it takes only 47-50 ms. That is an over 5 times performance gain.

B. Reduce Communication Cost

The cost of communicating an SPR from the (simulated) positioning technology to an LULS was also evaluated. The processing time of each rule was measured both when SPRs were sent from another host and then read through a local socket, and when SPRs are simply read from a local file.

The execution times of these two approaches are shown in Table 2. For rules that access database and therefore take a long time, the performance does not differ much by using the two approaches. However, for simple rules that do not access database, reading SPRs from a file can be 20-30 times faster than reading SPR from a socket. The reason is that reading SPRs from a socket does take more time than reading from a file, but for rules that access a database and take a long time itself to execute; this performance gain is not obvious since database access cost dominates the overall cost. What is learned from this experiment is that the communication cost can be reduced, probably by exploiting some caching mechanism to store the incoming and outgoing data, so that the server does not waste precious CPU cycle waiting for the data to be read in or sent out.

Identify the Most Time Consuming Functions

To find out which operations or functions take the most time during the rule evaluation, Rational Quantify, a runtime analysis tool, was run to help developers identify performance bottlenecks and portions of code subject to performance optimization. Rational Quantify was run for both rules that access the database and that do not. Some representative results are shown in Tables 3 and 4.

It is shown that:

- For rules that do not access database, the largest amount of time is taken by JVM Garbage Collector. As show in Table 4, the system spent one third of the time doing garbage collection.

Table 1. Execution time of the functions

Process Time		
Rules	Parameters	Time(ms)
ContainedIn.Polygon	Area 10	3.6
	Area 100	3.7
	Area 1000	3.7
ContainedIn.PolygonClass	100 polygons	3-4
	1000 polygons	11-13
	10000 polygons	28-32
	100000 polygons	270-300
DistanceFrom	Point (PointConstant)	0.37
	Point (PointVariable)	9.1
	Subscriber (SubscriberID)	0.37
HasIdentity	SubscriberID	0.38
HasValue	-	1.6
IsMemberOf	100 members	0.37
	1000 members	0.41
	10000 members	0.56
	100000 members	0.61
IsTrue/IsFalse	-	0.26
Logical	-	0.26
Relational	-	0.33
WithinDistanceOf.Point	-	9.7
WithinDistanceOf.PointClass	100 points	3.2
	1000 points	7.4
	10000 points	25-30
	100000 points	170-200
WithinDistanceOf.Subscriber		0.26
WithinDistanceOf.SubscriberClass	100 members	4.7
	1000 members	5.8
	10000 members	13.7

- For rules that access the database, the largest amount of time is taken by database operations. Table 4 shows that database related operations take more than half of the time and Java Garbage Collector takes another quarter.

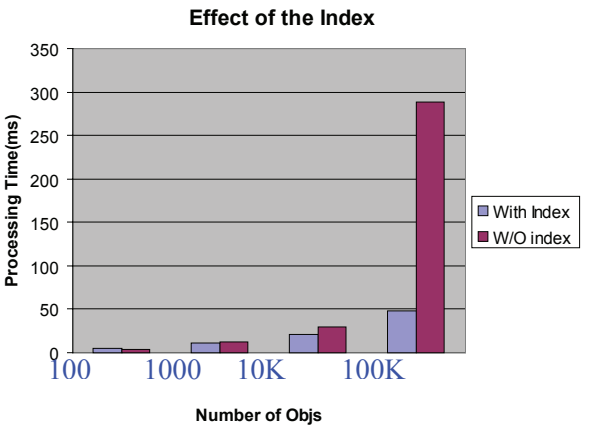
Both of these discoveries leave a great space for system optimization.

System Capacity Planning

Studying the execution time of each rule gives us an idea of the processing capability of an LULS. For example, give a certain number of subscribed users and rules and location update frequency, can an LULS process the entire load in time? Or for a given update frequency, how many subscribers or services can an LULS handle?

Let us suppose objects have a uniform update frequency and they update their locations every T

Figure 3. Effect of the index



seconds and the number of objects in the LULS is n. Suppose the location update of object i result in K_i (i= 1,…n) rules to be revaluated (assume other rules are irrelevant) and the processing time of the rules are t_{ij}, (i = 1, …, n; j = 1, …, K_i). Then the total processing time needed for each cycle is $\sum_{i=1}^{n}\sum_{j=1}^{K_i} t_{ij}$. For the LULS to handle the load in time, $\sum_{i=1}^{n}\sum_{j=1}^{K_i} t_{ij}$ should to be less than T. By studying the user profiles (how many services a typical user would subscribe to and how complicated these services are) and distribution, it can be estimated that the maximum number of users each LULS can handle for an update frequency. When $\sum_{i=1}^{n}\sum_{j=1}^{K_i} t_{ij} > T$, system administrators have to choose either evaluating rules at a lower frequency or do load shedding, which is a technique in data stream systems that discard some fraction of the unprocessed data in order for the system to continue to provide up-to-date query responses. Although both approaches could potentially harm the freshness of the queries, there are some heuristics that have been proposed earlier to maximize the accuracy.

OPTIMAL CELL SIZE FOR SERVER DEPLOYMENT

In order to achieve scalability, the system is designed in a distributed mode, where each local server covers a cell and handles the subscribers and applications related to that cell. A key issue in this environment is: what is the optimal cell size for deploying the local servers? Choosing the cell size or covering region of local servers suffers from two conflicting requirements. On the one hand, a small cell size is desirable since the smaller the cell is, the fewer the subscribers are in each cell and the faster the response to the location-based services could be. On the other hand, a large cell size is desirable since the smaller cell size can potentially result in a large number of crossovers between the cells. Each crossover requires a certain amount of overhead including rule migration, resources transferring, and rule activation or deactivations.

Table 2. Effect of the communication cost

SPR from Socket vs Local		
Rule	Socket	Local File
DistanceFrom.PointVariable	9.1	9.0
DistanceFrom.PointConstant	0.37	0.016
DistanceFrom.SubscriberID	0.37	0.015
IsTrue/IsFalse	0.26	0.010
HasIdentity	0.38	0.012
Logical	0.26	0.011
Relational	0.33	0.012

Table 3. Method time for the rule HasIdentity

Method&Calls	Method Time	Method time %
JVM Garbage Collector	7005.13	35.14
String.charAt	1321.40	6.63
LocalPerformanceTester.main	773.14	3.88
FloatingDecimal.readJavaFormatString	535.88	2.69

Table 4. Method time for the rule WithinDistanceOf

Method&Calls	Method Time	Method time %
DB2PreparedStatement.SQLExecute	61,816.39	47.04
JVM Garbage Collector	29,920.98	22.77
DB2ResultSet.SQLFreeStmtClose	12,434.50	9.46
FSRuleTable.processALL	1437.27	1.09
String.charAt	1348.51	1.03

Since the control center is involved in processing the handoffs, a large number of handoffs could make the control center become the bottleneck of the system. These conflicts are shown in Figure 4. By balancing these two factors, an approach was proposed to compute the optimal cell size for deploying the local servers.

Let us denote P(t) as the processing load of the local server during a time period t. Let α be the number of subscribed rules in the local server, λ be the subscribers' arrival rate and μ be the departure rate of subscribers, then $P(t) = K_1 \bullet \alpha \bullet t \bullet SPRrate + K_2 \bullet \beta \bullet t + K_3 \bullet \gamma \bullet t$.

The first part is the workload to handle all the subscribed rules in local server during the time t. Assume K_1 is the average cost of evaluating one rule and α is the number of subscribed rules, then $K_1 \bullet \alpha$ is the total cost of going through all rule evaluation once. $t \bullet SPRrate$ is the number of new position records received during this time period t. Since for each new SPR, local server needs go through all the rules, therefore, $K_1 \bullet \alpha \bullet t \bullet SPRrate$ is the cost of processing all the subscribed rules in local server during the time period t.

Besides processing the subscribed rules, the local server also has to process handoffs. Suppose K_2 is the cost of handling a subscriber arrival and K_3 is the cost of handling a subscriber leaving, β is the subscribers arrival rate and γ is the subscribers departure rate, then $K_2 \bullet \beta \bullet t + K_3 \bullet \gamma \bullet t$ is the total overhead to process arriving and departing users during this time segment. Since the control center is involved in handling handoffs, the time of waiting for the control center to process the handoffs should be measured. Assuming the arrival of events (handoffs) at the control center is a poisson distribution, the M/M/1 queuing model is used to compute the time taken by the control center. The average time of each event in the control center, including waiting time and service time, should be $\frac{1}{\mu - \lambda}$, where μ is the service rate and λ is the event arrival rate (Jose, Moreira, Rodrigues, & Davies, 2003). While μ is determined by the processing capacity of the control center, λ is highly related to the cell size. The smaller the cell is, the more handoff would occur. However, the exact relation between the cell size and the number of handoffs are determined by another factor—traffic.

According to Nanda (1993), with homogeneous traffic, the handoff rate is directly proportional to the total boundary length and it increases as the square-root of the increase in the cell density. An n^2 increase in the number of cells per unit area results in a handoff rate increase by a factor of n. With non-homogeneous traffic, traffic can be dominated by a few or just one traffic path; the increase in handoff rate is linear with the increase in cells per unit area. The optimal cell size for local servers in both cases is computed in the following sections.

Homogeneous Traffic

Under the assumptions that:

1. Mobile users are distributed uniformly in the region.
2. Direction of travel of mobile users is uniformly distributed over [0, 2π),

the handoff rate is proportional to the total boundary length (Nanda,1993). Thus, for each cell the number of handoff rate is Kr, where r stands for the edge length of the cell. Suppose the space is divided into n^2 cells, $n=\frac{l}{r}$, l is the side length of the whole space and r is the side length of a cell.

The handoff rate of the whole space should be:

$$n^2Kr = \frac{l^2}{r^2}Kr = K\frac{l^2}{r}.$$

The process load for each local server during time period t is:

$$P = K_1(c_1r^2)(t.SPRrate) + (+c)(K_2r)t = C_1r^2+(\frac{1}{\mu-\lambda}+c)C_2r$$

λ is the events arrival rate, which is the handoff rate in the system, as computed before, $\lambda = K\frac{l^2}{r}$.

$$P = C_1r^2+(\frac{1}{\mu-\lambda}+c)C_2r = C_1r^2+(\frac{1}{\mu-\frac{Kl^2}{r}}+c)C_2r =$$

$$C_1r^2+\frac{C_2r}{\mu-\frac{C_4}{r}}+cC_2r = C_1r^2+\frac{C_2r^2}{\mu r-C_4}+C_3r = C_1r^2+$$

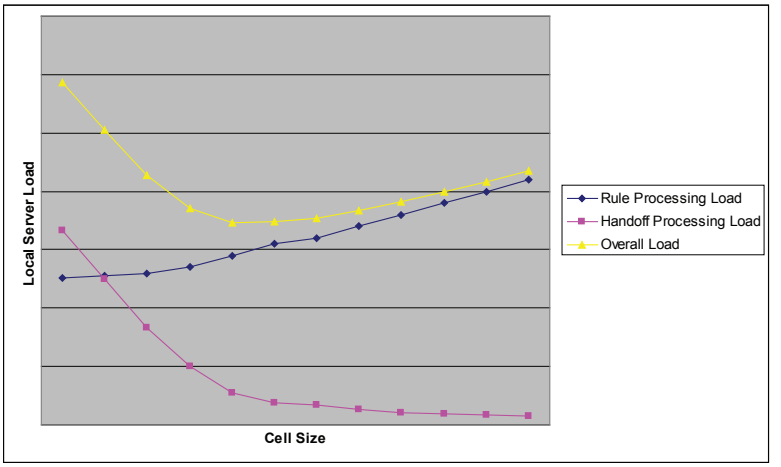
$$\frac{\frac{C_2}{\mu^2}(\mu r-C_4)(\mu r+C_4)+\frac{C_2C_4^2}{\mu^2}}{\mu r-C_4}+C_3r=C_1r^2+\frac{C_2}{\mu^2}(\mu r+C_4)$$

$$+\frac{\frac{C_2C_4^2}{\mu^2}}{\mu r-C_4}+C_3r = C_1r^2+\frac{C_2}{\mu}r+\frac{C_2C_4}{\mu^2}+\frac{\frac{C_2C_4^2}{\mu^2}}{\mu r-C_4}+C_3r$$

$$= C_1r^2+(\frac{C_2}{\mu}+C_3)r+\frac{C_2C_4}{\mu^2}+\frac{\frac{C_2C_4^2}{\mu^2}}{\mu r-C_4}$$

$$\frac{dP}{dr} = 2C_1r+(\frac{C_2}{\mu}+C_3)-\frac{C_2C_4^2}{\mu}(\mu r-C_4)^{-2}$$

Figure 4. Local server load



To get the minimum value of P let $\frac{dP}{dr} = 0$, then $2C_1 r + (\frac{C_2}{\mu} + C_3)\frac{C_2 C_4^2}{\mu}(\mu r - C_4)^{-2} = 0.$

This is a cubic equation that has general solutions, which is the optimal cell size. For the detailed solution, please refer to the appendix.

Non-Homogeneous Traffic

The result obtained in 3.1 is based on the assumption of the homogeneity of vehicular traffic. It applies to situations where each cell consists of a large number of traffic paths with random orientation. As cells get smaller, the traffic is no longer homogenous. The distinct paths in each cell, their orientation, and traffic flow become important. In this case, the increase in over handoff rate is linear with the increase in cell density (Nanda, 1993). Therefore, the handoff rate for each cell is basically a constant K. Suppose the space is divided into n^2 cells, $n=\frac{l}{r}$, l is the side length of the whole space and r is the side length of a cell. The handoff rate of the whole space should be $K n^2 = K\frac{l^2}{r^2}$.

Again, the process load for each local server during time period t is:

$$P = K_1(c_1 r^2)(t.SPRrate) + (\frac{1}{\mu - \lambda} + c) K_2 t = C_1 r^2 + (\frac{1}{\mu - \lambda} + c)C_2$$

λ is the handoff rate in the system, which is, as computed before, $K\frac{l^2}{r^2}$.

$$P = C_1 r^2 + (\frac{1}{\mu - \lambda} + c)C_2 = C_1 r^2 + (\frac{1}{\mu - K(\frac{l}{r})^2} + c)C_2 =$$

$$C_1 r^2 + \frac{C_2}{\mu - \frac{Kl^2}{r^2}} + C_3 = C_1 r^2 + \frac{C_2 r^2}{\mu r^2 - Kl^2} + C_3 =$$

$$C_1 r^2 + \frac{\frac{C_2}{\mu}(\mu r^2 - Kl^2) + \frac{C_2^2 l^2}{\mu}}{\mu r^2 - Kl^2} + C_3 = C_1 r^2 + C_4 +$$

$$\frac{C_5}{\mu r^2 - Kl^2} + C_3$$

Take r^2 as a variable X, the above is $C_1 X + \frac{C_5}{\mu X - Kl^2} + (C_3 + C_4)$

$$\frac{dP}{dX} = C_1 - C_5\mu(\mu X - Kl^2)^{-2}$$

to get the minimum value of P, let $\frac{dP}{dr} = 0$, we get $(\mu X - Kl^2)^2 = \frac{C_5\mu}{C_1}(\mu X - Kl^2)$ should be larger than 0, for the reason that $(\mu X - Kl^2) = X(\mu - K\frac{l^2}{X})$ and X, which is r^2, is positive; and $\mu - K\frac{l^2}{X} = \mu - K(\frac{l}{r})^2$ is the process time of control center, which is also positive.

Therefore, $\mu X - Kl^2 = \sqrt{\frac{C_5\mu}{C_1}}$, which means $X = \frac{\sqrt{\frac{C_5\mu}{C_1}} + Kl^2}{\mu}$

Since $C_5 = \frac{C_2^2 l^2}{\mu}$, $X = \frac{\frac{C_2 L}{\sqrt{C_1}} + Kl^2}{\mu}$.

Hence, the optimal cell size is proportional to the handoff rate and to inverse of the control center service rate and the residents' rule processing overhead, that is, the higher the hand off rate is, the larger the cell should be. The more time needed to process the residents' subscription rules, the smaller the cell should be. The smaller the process rate of the control center is, the larger the cell should be.

CONCLUSION

LBS is surely an area of modern mobile services where considerable growth is observed. The developments in the Internet domain, wireless/mobile networking, as well as the proliferation of positioning technologies expedited such evolution. The impact on nomadic users is tremendous. It is evident that such progress needs to be addressed in a methodological manner and supported, where appropriate by coordinated standardization efforts. Requirements need to be reviewed and studied

very carefully by all the involved actors. Their coverage—fulfillment by modern technological platforms, is also a very important issue. Our work is mainly moving along these lines. The analysis and design techniques, as presented in this chapter, shows that the technologies and issues involved in LBS deployment and provision cover a very wide spectrum including operating system capabilities, user interface design, positioning techniques, terminal technologies, network capabilities, and so forth. The meticulous mapping of these technical aspects to the identified requirements is a critical success factor for LBS.

REFERENCES

Agre, J., Akinyemi, A., Ji, L., Masuoka, R., & Thakkar, P. (2002). *A layered architecture for location-based services in wireless ad hoc networks*. Retrieved on August 15, 2007, from www.flacp.fujitsulabs.com/~rmasuoka/papers/200203-LocationProtocol7.doc

Chen, X., Zhang, F., Sun, M., & Luo, Y. (2004). System architecture of LBS based on spatial information integration. *Geoscience and Remote Sensing Symposium, IGARSS IEEE International, 4*, 2409-2411.

Hermann, F., & Heidmann, F. (2002). User requirement analysis and interface conception for a mobile, location-based fair guide. Mobile HCI, Lecture Notes in Computer Science, 2411, 388-392.

Hightower, J., & Borriello, G. (2001). Location systems for ubiquitous computing. *IEEE Computer, 34*(8), 57-66.

Jacob, K. (2007). *Location based services*. Retrieved on May 15, 2007, from http://www.kenneyjacob.com/2007/05/13/location-based-services/

Jose, R., Moreira, A., Rodrigues, H., & Davies, N. (2003). The AROUND architecture for dynamic location-based services. *Mobile Networks and Applications, 8*(4), 377-387.

Munson, J., Cole, A., Duri, S., & Wood, D. (2003). *Architectural specification of the location utility* (Version: 0.4). IBM report.

Nanda, S. (1993). Teletraffic models for urban and suburban microcells: Cell sizes and handoff rates. *IEEE Transaction of Vehicular Technology, 42*(4).

Saltenis, S., Jensen, C., Leutenegger, S., & Lopez, M. (2000). Indexing the positions of continuously moving objects. SIGMOD, 331-342.

Sistla, A. P., Wolfson, O., Chamberlain, S., & Dao, S. (1997). Modeling and querying moving objects. *ICDE,* 422-432

Steiniger, S., Neun, M., & Edwardes, A. (2007). *Foundations of location based services*. Retrieved August 26, 2007, from http://www.spatial.cs.umn.edu/CS8715/IM7_steiniger.pdf

Tao, Y., Papadias, D., & Sun, J. (2003). The TPR*-Tree: An optimized spatio-temporal access method for predictive queries. VLDB, 790-801.

Tsalgatidou, A., Veijalainen, J., Markkula, J., Katasonov, A., & Hadjiefthymiades, S. (2003). Mobile e-commerce and location-based services: Technology and requirements. *ScanGIS,* 1-14.

Virrantaus, K., Markkula, J., Garmash, A., & Terziyan, Y. V. (2001). Developing GIS-supported location- based services. In *Proceedings of the WGIS'2001—First International Workshop on Web Geographical Information Systems* (pp. 423-432). Kyoto, Japan.

Wolfson, O., Xu, B., Chamberlain, S., & Jiang, L. (1998). Moving objects databases: Issues and solutions. *Statistical and Scientific Database Management,* 111-122.

KEY TERMS

Location Based Services: Location based services are information services accessible with

mobile devices through the mobile network and utilizing the ability to make use of the location of the mobile device (Virrantaus et al., 2001).

Location Based Access: the user can access map and catalogue data based on his/her present location. If locating the user cannot be resolved automatically by the system, the user must be provided with the option of manually entering his/her location. The system should be able to handle such requests through geocoding procedures.

Location Utility (LU): Location utility is a distributed LBS system proposed and developed by the IBM Watson Research Center. It is developed for supporting a large class of applications that involve responding to the movements of large numbers of mobile users. It consists of a control center (LUCC) and a set of local servers (LULS). This distributed architecture is proposed for the purpose of scalablity. Application rules/functions are distributed to all local server nodes, and each local server handles the subset of the users associated with the local node (Munson et al., 2003).

Location Utility Local Server (LULS): LULS nodes receive the raw subscriber position information from the positioning technology and evaluate the rules assigned to them. Rules that evaluate to true generate reports, which are sent to the application that subscribed to the rule. Where possible, local server nodes are deployed in alignment with the network infrastructure in order to take advantage of local information about the user presence (Munson et al., 2003).

Location Utility Control Center (LUCC): The LUCC is the locus of control and management functions, and is the primary point of contact for applications to subscribe to rules and set up application resources. It also provides an interface for retrieving subscriber positions and any resources applications may have created for them. When an application subscribes to a rule through the control center, the control center records the subscription information and forwards the rule to all local server nodes. Reports from these rules are sent from the local server nodes to the control center, which aggregates them and forwards them to the application (Munson et al., 2003).

Moving Objects Spatio-Temporal (MOST) Model: MOST model was proposed for databases with dynamic attributes, for example, the positions of moving objects. The dynamic attributes of objects can change continuously as a function of time, without being explicitly updated. Therefore, the answer to a query depends not only on the database contents, but also on the time at which the query is entered (Sistla et al., 1997).

Positioning Determination Technologies (PDT): PDT are technologies that exist for determining physical position. The user position can be obtained either by using the mobile communication network or by using the global positioning system (GPS). Further possibilities to determine the position are WLAN stations, active badges, or radio beacons. The latter positioning methods can be especially used for indoor navigation like in a museum.

Spatial Indexes: Spatial indexes are used by spatial databases to optimize spatial queries. Spatial indexes can effectively handle features such as how far two points differ and whether points fall within a spatial area of interest. Common spatial index methods include Grid, Quadtree, R-tree, and kd-tree.

APPENDIX

Find the General Solution to:

$$2C_1 r + \left(\frac{C_2}{\mu} + C_3 \right) = \frac{C_2 C_4^{\,2}}{\mu} (\mu r - C_4)^{-2}$$

Take $\mu r - C_4 = X$, and let both side of the equation multiply by μX^2,

$$2C_1 (X + C_4) X^2 + (C_2 + C_3 \mu) X^2 = C_2 C_4^{\,2}$$

$$2C_1 X^3 + (2C_1 C_4 + C_2 + C_3 \mu) X^2 - C_2 C_4^{\,2} = 0$$

$AX^3+BX^2-C=0$

We apply the substitution $X= Y-\frac{B}{3A}$ to the cubic equation and obtain:

$$A(Y-\frac{B}{3A})^3+B(Y-\frac{B}{3A})^2+C=0$$

$$AY^3-\frac{B^2}{3A}Y+(C+\frac{2B^3}{27A^2})=0$$

We are left with solving a cubic equation of the form $y^3+Py=Q$. How to solve this equation had been discovered earlier by *Scipione dal Ferro*.

We will find s and t so that:

$3st = P$ (1)

$s^3-t^3=Q$ (2)

It turns out that $y=s-t$ will be a solution. Let's check that: Replacing A, B and y as indicated transforms our equation into:

$(s-t)^3+3st\,(s-t)=s^3-t^3$.

This is true since we can simplify the left side by using the binomial formula to:

$(s^3-3s^2t+3st^2-t^3)+(3s^2t-3st^2)=s^3-t^3$.

Solving the first equation for s and substituting into (2) yields:

$$(\frac{P}{3t})^3-t^3=Q$$

Simplifying, this turns into the "tri-quadratic" equation:

$$t^6+Qt^3-\frac{P^3}{27}=0$$

$$u^2+Qu-\frac{P^3}{27}=0$$

which using the substitution $u=t^3$ becomes the quadratic equation:

$$u^2+Qu-\frac{P^3}{27}=0$$

From this, we can find a value for u by the quadratic formula,

$$u=\frac{-Q\pm\sqrt{Q^2-\frac{4P^3}{27}}}{2}$$

then obtain t,

$$t=\sqrt[3]{\frac{-Q\pm\sqrt{Q^2-\frac{4P^3}{27}}}{2}}$$

By equation (2),

$$s^3=Q+\frac{-Q\pm\sqrt{Q^2-\frac{4P^3}{27}}}{2}$$

Y is the difference of s and t:

$$Y=\sqrt[3]{Q+\frac{-Q\pm\sqrt{Q^2-\frac{4P^3}{27}}}{2}}-\sqrt[3]{\frac{-Q\pm\sqrt{Q^2-\frac{4P^3}{27}}}{2}}$$

The solution to our original cubic equation is given by:

$$X=Y-\frac{B}{3A}=$$

$$\sqrt[3]{Q+\frac{-Q\pm\sqrt{Q^2-\frac{4P^3}{27}}}{2}}-\sqrt[3]{\frac{-Q\pm\sqrt{Q^2-\frac{4P^3}{27}}}{2}}-\frac{B}{3A}$$