LORE: An infrastructure to support location-aware services

Y. Chen
X. Y. Chen
F. Y. Rao
X. L. Yu
Y. Li
D. Liu

With the advance in wireless Internet and mobile computing, location-based services (LBS)—the capability to deliver location-aware content to subscribers on the basis of the positioning capability of the wireless infrastructure—are emerging as key value-added services that telecom operators can offer. To support efficient and effective development and deployment of innovative location-aware applications, a flexible and resilient middleware should be built as the enabling infrastructure. This paper presents the research and efforts made in the IBM China Research Laboratory toward developing an infrastructure that supports location-aware services. This infrastructure is based on a proposed location operating reference model (LORE), which addresses many major aspects of building location-aware services, including positioning, location modeling, location-dependent query processing, tracking, and intelligent location-aware message notification. Three key components of the infrastructure—the location server, a moving object database, and a spatial publish/subscribe engine—are introduced in detail. The location server has a common location adapter framework that supports heterogeneous positioning techniques and industry-standard location application program interfaces (APIs). The moving object database manages the location stream and processes the location-based queries. The spatial publish/subscribe engine enables intelligent location-aware message notification. We also present some location-aware application demonstrations that leverage the LORE infrastructure. Part of our work has been tested in pilot projects with leading carriers in China and has been integrated into the IBM WebSphere® Everyplace® Suite.

Introduction

With advances in wireless Internet and mobile computing, location-based services (LBS) are emerging as key value-added services for telecom operators to deliver. LBS enables them to provide personalized location-aware content to subscribers using their wireless network infrastructure. Besides telecom operators, more and more service providers such as public wireless LAN providers, enterprises, and others are developing and deploying location-aware services for consumers and employees to gain more revenue and productivity. These location-aware service providers are facing both technical and social challenges, such as positioning in various environments using different locating mechanisms, location tracking, the information delivery model, privacy issues, and the development of innovative LBS applications that succeed at delivering more business impact and value. It has been realized that a flexible and resilient middleware should be

0018-8646/04/$5.00 © 2004 IBM

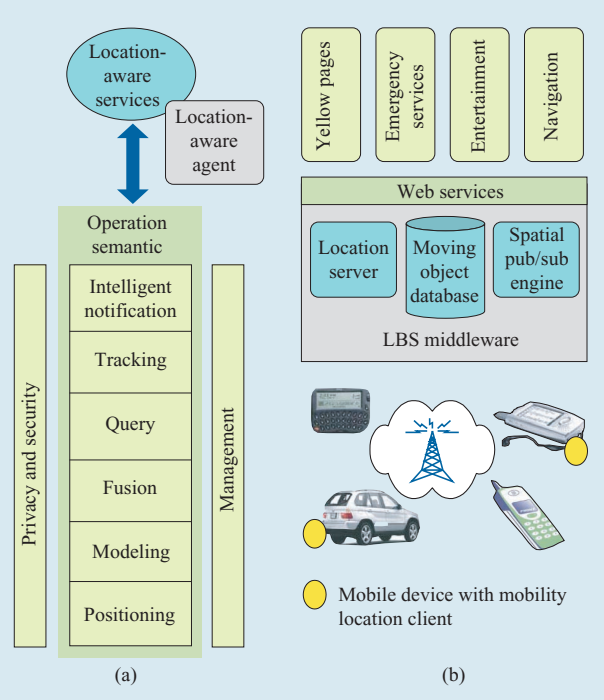

Figure 1

(a) Location operating reference model. (b) Infrastructure supporting location-aware services.

built as the enabling infrastructure to support the different players, so that a service provider can efficiently, effectively, and quickly develop and deploy LBS applications and support innovative location-aware applications.

The location-aware infrastructure should address the key challenges in location-aware computing identified in [1]. These include technology-independent location sensing, end-to-end control of location information, tracking and predication, and other research challenges involving geospatial information processing and human interaction with this information.

We address the challenges listed above in an infrastructure supporting location-aware services. A location operating reference model (LORE) is built to capture the location operation semantics by multiple layers in which a richer location operation semantic is modeled at a higher layer. The location operation semantics we present address many issues—for example, how to retrieve the location data, how the location data is modeled, how to fuse location from different location sources, how to query the location data, how to use a tracking mechanism to deliver intelligent location-aware notification, and so on. In addition to the semantics, two other important dimensions in location-aware computing—privacy protection and management—are also covered by the LORE model. On the basis of the LORE model, different components of the location-aware infrastructure are built to meet the requirements of different layers and expose application program interfaces (APIs) to developers who can then build other components that can plug into the model. In the following sections, several key components of the LORE infrastructure are introduced to show issues of the location-aware computing we address and to explain how the composition of components could facilitate the development of various location-aware services.

The paper is organized as follows. In the next section, the LORE model and the infrastructure are presented. The three key components of the infrastructure—the location server with common adapter framework, the moving object database, and the spatial publish/subscribe (pub/sub) engine—are introduced in the subsequent sections. A location-aware messaging prototype that leverages the infrastructure is then presented. After that, we discuss related work, and then summarize our work and discuss our future direction.

Location operating reference model and infrastructure

Figure 1(a) illustrates the proposed LORE model to capture the semantics and management issues required when building location-aware services. The LORE model includes four domains: operation semantic, management, privacy and security, and location-aware agent.

Operation semantic domain

The operation semantic domain includes layered components which are, from bottom to top: positioning, modeling, fusion, query, tracking, and intelligent notification. The layered components explicitly describe the dependencies among components; i.e., the upper component uses the functionalities exposed by lower components to build more advanced functionalities. The overall functionalities provide the capabilities for location-aware applications requiring a rich location operation semantic.

The positioning component addresses the issue of technology-independent location sensing, i.e., how to obtain the location information of a target object via specific positioning mechanisms. Technical neutral positioning requires that the positioning component be able to interface with heterogeneous positioning equipment and expose a uniform virtual positioning mechanism for other components. The component has to deal with two different modes of positioning: server-based and client-based. In server-based mode, the location of the target object is measured and calculated on the server side. For example, Global System for Mobile Communications** (GSM**) networks could determine

the subscriber's position by the cell in which the mobile phone is being served. In client-based mode, the device does self-positioning; e.g., the global positioning system (GPS) device can determine its location. The major difference between the two modes is the way in which the positioning component retrieves the location information. In server-based mode, the component pulls the location from the server by accessing the location interface—e.g., the Location Interoperability Forum (LIF) [2] interface—exposed by the server. In client-based mode, the device always pushes the location to the positioning component, because it is difficult for the client to have a location interface. Two positioning modes require the positioning component to support both push and pull models.

The modeling component describes the semantic of location information. Because location data can come from different positioning mechanisms, it shows great heterogeneities in syntax, name, type, and metric system. For example, the LIF exposes location data in eXtensible Markup Language (XML) format, while GPS exposes location data in compact binary format. GPS can provide velocity information, while most GSM positioning approaches could not provide such data. The modeling component integrates the heterogeneous location data by providing a uniform location model that facilitates the development of flexible services. The location model captures sufficient information on location, including coordinates, time, velocity, error, and other related information.

The fusion component addresses the issue of how to derive an accurate location by fusing location reports from multiple devices for one target object. For example, a person has a cell phone, a notebook computer with a wireless card, and a GPS receiver. All of these devices can be positioned, and their location reports are sent to the fusion component to determine the precise location. The fusion component derives the precise location on the basis of a predefined rule set, which may define the possibilities of location accuracy in different contexts. There are many interesting topics in the location fusion algorithms and rule set to be researched. Among the research work in this area, Myllymaki and Edlund [3] have proposed a methodology for aggregating location data from multiple sources associated with a moving object.

The query component provides spatio-temporal query interfaces from which applications or end users could obtain location information for interested objects and issue location-related queries. The query could involve not only current location information, but also historical and/or future location information. A typical location query is *Please report the location of object X*. Another, more complex, spatio-temporal query involving historical information is *Please report the objects that are in zone X at time Y*. The query component depends on positioning and/or fusion components to obtain the location data. To support effective historical and current location information retrieval, the query component employs a spatial index to improve the query performance. The spatial index could be an R-tree and its variations, grid index, Z-order, and so on. Some location-predication mechanism could be used by the query component to answer the question about the future location of specific objects.

The tracking component plays a key role in location-based services in the sense that most LBS applications require the tracking locations of target objects so that they can determine the trajectory and provide information based on the location or trajectory of the target object. Typical applications include fleet management, taxi dispatch, and road assistance navigation. Tracking puts significant performance impact on the underlying positioning component by positioning the location of the objects continuously or in a specified time interval.

The intelligent notification component sends location-dependent messages, including sales promotion, weather and traffic information, nearby events, and so on. A typical application of the intelligent notification is *Please send me promotion message while I am in zone X*. The key technology behind intelligent notification is spatial pub/sub service. Users define, in advance, the events in which they are interested by specifying spatio-temporal conditions. Then, by taking user location information into consideration, notification is delivered to them when the conditions are met. While the intelligent notification component is deployed to support a large number of users, the spatial pub/sub should also provide a scalable mechanism to enable intelligent location-aware services.

Management domain

The management domain includes the mechanisms necessary to support managing the components in the operation semantic domain. With the exception of privacy and security issues, these components include configuration management, policy management, monitoring and logging, component availability, and so on.

Privacy and security domain

Privacy and security play an important role in building location-aware business services, because a user's location is an important aspect of privacy information and should be protected from invalid use and exposure. The privacy and security domain provides a flexible framework to guarantee that the use of location information is under control in the location-aware services environment. In the privacy framework, the user can decide who or which service has permission to obtain location information; furthermore, the user can define where, when, and why (for what purpose) the information can be retrieved

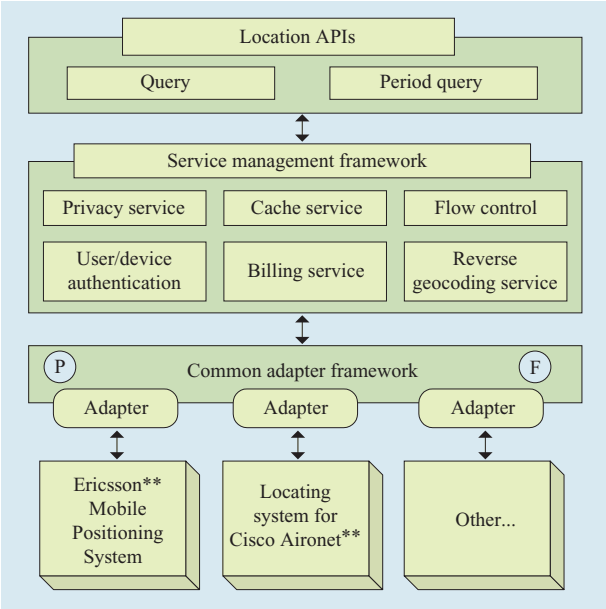

Figure 2

Location server architecture. [The circled "P" indicates connection pools and a mobile station identifier (MSID) combination. The circled "F" indicates fusion algorithms.]

or used. The security framework protects location information by leveraging proven security mechanisms, such as encryption, digital signature, and secure transportation protocol.

Agent domain

Advances in mobile computing mean that mobile devices such as cell phones and personal digital assistants (PDAs) gain more computing, networking, and storage capability. More scalable location-aware services and innovative user experiences can be built by taking advantage of the resources in such devices. The agent domain introduces the location-aware agent that resides in the mobile device and cooperates with servers to complete location-aware services. For example, in the tracking service, the self-positioning client could send the location information to the server only when changes in the location are larger than a predefined threshold. This can reduce network traffic and server resource consumption. The agent domain provides a framework to build a service-specific location-aware agent.

Infrastructure supporting the LORE model

On the basis of the LORE model, the infrastructure that supports location-aware services is proposed as shown in **Figure 1(b)**. With the support of the infrastructure, location-aware services such as yellow pages, emergency services, entertainment, and navigation services could easily be created and deployed. Three prototypes of the key components in the LBS middleware of the infrastructure are implemented, and all components in the LORE operation semantic domain are covered.

Location server

The location server (LS) provides the positioning, modeling, and fusion components in the LORE operation semantic domain. It also supports simple query and tracking functions. The *common adapter framework* is introduced to support technology-independent location sensing, which is detailed in the next section. The location server supports the Wireless Application protocol (WAP) [4] location API and LIF [2] interface for retrieving and querying location information. Also, the location server includes a privacy mechanism to protect location information from being used without the owner's permission.

Moving object database

The moving object database (MOD) manages the location data collected from the location server and provides the query and tracking components in the LORE operation semantic domain. The *continuous, active monitoring engine for location-based services* (CAMEL) [5] is built as a MOD prototype that supports queries of both historical and current location information. MOD is discussed in the section on moving object databases.

Spatial pub/sub engine

The spatial pub/sub engine supports the intelligent notification component in the LORE operation semantic domain. It provides interfaces for subscribing location-aware messages and defining system-wide or application-specific location information for subscription. A section below is devoted to describing the spatial pub/sub engine.

In addition to the LBS middleware, the infrastructure also includes a mobility location client (MLC) that supports the location-aware agent domain in the LORE model. An MLC framework based on the Java** 2 Platform, Micro Edition (J2ME**) is implemented and supports the JSR179 specification (Location API for J2ME) [6]. The MLC enables the applications in mobile devices to leverage local resources and cooperate with remote servers to improve system performance and reduce network traffic.

Location server

The location server provides the positioning, modeling, and fusion components of the LORE operation semantic domain and supports the privacy and security and management domains in the LORE model. The architecture of the location server is depicted in **Figure 2**.

The location server has three layers: location APIs, the service management framework, and the common adapter framework. It is designed with three features in mind: flexibility in location API, scalability in the common adapter framework, and extensibility in service management.

Common adapter framework

The common adapter framework (CAF) provides standard APIs to fetch the location information of the target object without regard to the positioning mechanisms. It defines a common adapter interface intended to shield the details of various positioning systems and provides one adapter implementing this interface for each underlying positioning system. Each vendor-specific adapter focuses on dealing only with transport—HyperText Transfer protocol (HTTP) or Transmission Control protocol (TCP)—and the XML format transcoding. A new adapter can easily be developed and plugged into the framework in a short time.

In some specified infrastructures [for example, in GSM code division multiple access (CDMA) wireless networks], positioning is a resource- and time-consuming process. CAF provides performance optimization mechanisms such as connection pools and a mobile station identifier (MSID) combination, specified as a circled *P* in Figure 2. Connection pools are designed to reuse socket connections and restrict the maximum network traffic. An MSID combination can bind multiple concurrent location queries to one location query, so that the corresponding adapter talks to the positioning equipment only once and obtains location information for multiple MSIDs.

CAF supports multiple adapters simultaneously by various fusion algorithms, specified as a circled *F* in Figure 2. There are many kinds of policies to retrieve location from multiple adapters. The simple way is a brute force solution in which every request is permitted to go to all adapters and the proper one is selected. Another way is to order all adapters according to their probabilities calculated from the history data. The fusion algorithm from multiple adapters is still a research challenge.

Service management framework

The goal of the service management framework is to address the issues in the privacy and security and management domains of the LORE. These services implement the common service interfaces defined by the framework and, as the framework is currently configured, can be called before or after the location acquisition. The services currently supported by the framework are privacy, user/device (U/D) authentication services, cache, flow control, billing, and reverse geocoding. New services can be developed and plugged into the framework on the basis of the interface.

Since privacy control based on user and location acquisition is from the device, the U/D pair should be completely authenticated in order to avoid potential disclosure of privacy information. *U/D authentication service* provides U/D mapping information according to the open U/D authentication API defined in the system. Note that there is no concrete implementation for U/D authentication service in the location server. The service is implemented in a domain-specific or solution-specific way. For example, it can be implemented on the basis of the enterprise employee database for an enterprise location-aware system, or it can be based on the user profile repository for mobile operators who deploy LBS applications.

Because the indiscriminate use of location information for people can infringe on personal privacy, fine-grained access control to location information is necessary. *Privacy service* provides a privacy protection mechanism based on the role-based access control model, with time and location constraints. Users can determine who can access what location information under which circumstances.

Because location acquisition is a time- and resource-consuming process, *cache service* was introduced to accelerate responsiveness. The goal of cache service is to maximize the use of available location information under the caching strategy, thus reducing the consumption of system resources and improving performance.

Flow control service is designed to avoid location server traffic congestion and to ensure fair play among applications. There are two kinds of constraints: *application-independent constraint*, such as the maximum concurrence requests limit, and *application-dependent constraint*, such as the maximum number of requests allowed within a given time period and the minimum interval among consecutive successful requests. By supporting effective and efficient flow control, the location server can avoid attacks resembling denial of service and resource overspending.

Billing service is a special logging service to facilitate billing for location services. It logs detailed request/response data into output files from which necessary information can be extracted by various billing engines to generate billing reports.

Reverse geocoding defines the interface to map the raw location data to a normalized and meaningful symbolic address such as *street*, *city*, or *zip code*. Consequently, there are two types of reverse geocoding. One is a common process that provides domain-independent reverse geocoding, for example, at the city or country level, and another is an application-specific process that provides domain-dependent reverse geocoding, such as for an enterprise or office building. The implementation for this service can be either self-developed or a wrapper for third-party reverse-geocoding modules.

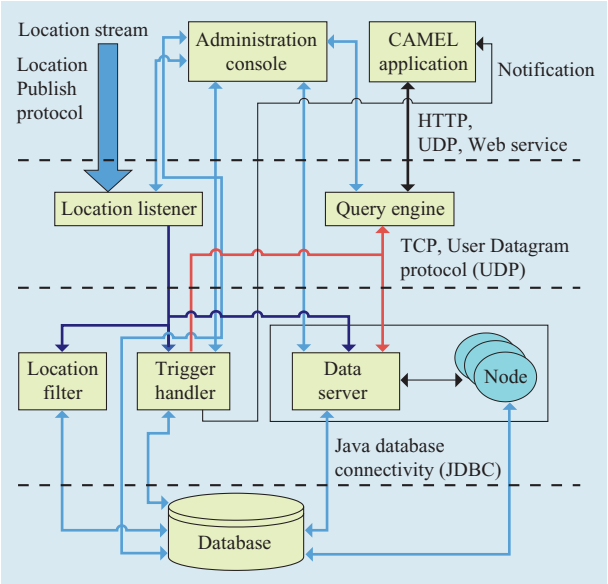

Figure 3

CAMEL, the moving object database architecture.

Location APIs

The location server addresses the modeling issue by defining a common open location model that includes geolocation, address, timestamp, and application-specific information. The location information based on this model adapts to various positioning techniques and covers all information required by applications.

Two kinds of query modes are supported: *query* and *subscription*. In query mode, the location of a target object can be sent to the requester immediately. In subscription mode, the locations of target objects are sent to the requester at specified intervals. Two sets of primitive messages are defined for the modes, respectively: the *query service primitive messages set* and the *subscribe service primitive messages set*. Each set includes several primitive messages that describe the interaction pattern between the requestor and the requestee (location server). On the basis of the two core services and the corresponding primitive messages, it is easy to support different industry-standard location APIs, such as WAP and LIF, by mapping them to core services at the location server. For example, WAP immediate query service and deferred query service are respectively mapped to query service and subscribe service.

Moving object databases

CAMEL is a high-performance engine that manages the location stream to support query, tracking, and intelligent notification components in the LORE operation semantic domain. These components are typically developed to meet the requirements of next-generation intelligent location-aware services. CAMEL takes a MOD approach that not only stores historical and current location information of mobile users, but also predicts their future locations. The historical information captured in CAMEL can be used by a data mining tool to discover mobility patterns. The overall architecture for CAMEL is illustrated in **Figure 3**. CAMEL is composed of several components that can be physically distributed in different network nodes, which communicate with each other using standard protocols and interfaces such as TCP/IP, HTTP, and Java database connectivity (JDBC). CAMEL components include the location listener, query engine, location filter, trigger handler, data server, database, and administration console. The distributed-component-based architecture not only makes the system robust, but also facilitates the easy deployment of CAMEL in different environments. In this section, we briefly introduce the components.

Database

The database (DB) component is the heart of CAMEL and serves not only as storage for the location data, but also as the registry and configuration repository. The registry is used by the system to record component running information, such as host address, port number, and running status. When starting up, each component registers its host address and port information in the registry, and other dependent components use this information to find it. The configuration repository is a central repository for all components to store component-specific parameters. The registry and configuration repositories make the system more flexible and manageable.

Location data for each moving object at checkpoint time is stored in an object checkpoint table (OCT) in the database. OCT records the historical information of a moving object (*oid:* object identifier) and is used for historical queries and data mining. For example, the query *What is the trajectory of object A from time* t_1 *to* t_2*?* can be answered by the function,

$$trajectory\ (select\ location\ from\ OCT\ where\ oid = A\ and\ t_1 \le t \le t_2).$$

Location listener

The location listener (LL) accepts location reports from reporters, such as the tracking server or self-positioning devices, using a Location Publish protocol (LPP) based on HTTP. Any authenticated reporter can send location reports to LL via LPP. For performance reasons, a location report protocol that uses Java object serialization over User Datagram protocol (UDP) is also supported

by LL and facilitates Java-based CAMEL applications. Upon receiving a location report, LL propagates it using Internet protocol (IP) multicast to other registered location receivers that have registered themselves with LL at start-up. Potential location receivers are location filter, trigger handler, and data server. The presence of LL makes it easy to add new components (location receivers) to the system and to prevent incomplete or invalid location data from entering the system.

Query engine

The query engine (QE) is the main interface to issue queries about moving objects. Currently, the following types of queries are supported:

- *GetLocation:* Obtain the location of an object at a specified time.
- *Window query:* Obtain objects that are within a specified distance from a specified object.
- *kNN:* Obtain the *k* nearest objects relative to a specified object.
- *Trigger:* Send a notification when the location of a specified object meets a predefined condition.
- *Historical query*.

The query engine exposes its interface using Web services technology, and the query supported is represented by Web Services Description Language. QE forwards some type of query using TCP to underlying components, such as the trigger handler and data server.

Location filter

A high arrival rate of location reports from location reporters introduces two difficulties: the database insertion of location data may become a bottleneck, and there is a possibility of redundant location data. The location filter (LF) is designed to filter incoming location data to reduce the location stream while ensuring its quality. The LF is implemented as a location receiver of LL and writes the filtered location into the OCT database table. CAMEL implements several filter algorithms that typically reduce the original location stream by 60 to 80% while maintaining reasonable location accuracy.

Trigger handler

A spatial trigger is a very important kind of query in a MOD and is the base for push-based services in LBS. In CAMEL, trigger *Tr* is defined by a tuple *Tr = (SP, Sink)*, where *SP* is a spatial predicate with location variables and *Sink* is the access point of the notification receiver when the trigger is fired. Currently, four *Sink* types are supported: HTTP, TCP, UDP, and Console. They print only the notification on the standard output. When a location arrives, the trigger handler (TH) evaluates it against the triggers defined in the system (binding the location to the predicate) to check whether it satisfies any trigger predicate. If one trigger is satisfied, the notification, along with the triggered location, is sent to the sink defined in the trigger. The TH is implemented as a location receiver of the LL and supports the trigger operations `create trigger` and `drop trigger`. In the section below on the spatial pub/sub engine, we discuss spatial triggers in more detail, and how they can support the spatial pub/sub engine.

Data server

As mentioned above, CAMEL supports historical, current, and future location queries. In practice, most queries are related to the current location of objects (for example, kNN and window queries) and require real-time processing. The data server (DS) is responsible for processing only those queries that concern current locations (the most common query in location-aware services). To improve the performance of the query related to current location and avoid access to a database, a main memory snapshot of moving objects is used to manage current locations. More specifically, the current snapshot stores the latest locations from reporters instead of the current locations of moving objects. The snapshot is organized as a spatial index that accelerates the processing of some queries, such as the window query and the kNN query. Meanwhile, the spatial index is required to sustain high-performance updates and lookups because of the high arrival rate of location data. After comparing several spatial index methods, making a tradeoff between updating and searching, and taking the main memory characteristic into consideration, CAMEL employs a grid index as its snapshot indexing mechanism. The primary reason for this is that the index scales up well under high update rates and has a clean implementation with good performance when searching in main memory. This is in contrast to an R-tree, whose variations suffer from frequent index entry splitting and merging. Kwon et al. [7] proposed a lazy-update R-tree (LUR-tree) to reduce updates in the R-tree. It is, however, based on a continuous model that requires a predefined moving pattern such as speed.

One consideration is whether CAMEL will be set up in a region with a large area, or in several small administration regions. An example of the latter is Beijing, a large city composed of several districts. For scalability and load balancing, the DS is designed with a distributed architecture in which new components, *nodes*, are added to handle the moving objects that belong to each region. The use of nodes introduces new issues such as distributed query processing and mobile object migration/roaming. New mechanisms, such as the distributed query manager and roaming managers, are also introduced in the DS.

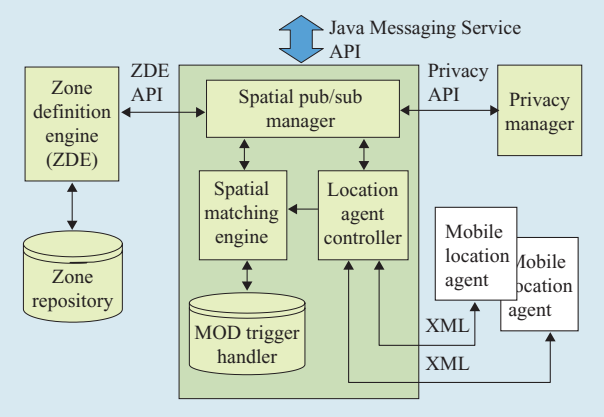

Figure 4

Spatial pub/sub engine architecture.

Administration console

The administration console is a text interface console used to configure and monitor other system components. The console reads runtime information from the database and connects to the various components. In addition, the user can use the console both to issue queries and as a testing tool.

Spatial pub/sub engine

With the support of a location server and a moving object database, location-based services to support applications such as location-aware yellow pages, road assistance, and tracking could be delivered. A spatial pub/sub engine brings a rich location operation semantic and provides new user experiences by enabling active push-based messaging on a user's subscription. In this section, the spatial pub/sub engine to support the intelligent notification component of the LORE model is described. The architecture of the engine is illustrated in **Figure 4**. The engine adopts a novel client-side event processing approach to improve performance by eliminating server-side computing cost. The mobile location agent, cooperating with the location agent controller, implements the role of the location-aware agent in the LORE model. The key components include the spatial pub/sub manager, the spatial matching engine, the zone definition engine, the location agent controller, and the mobile location agent.

Models

The models adopted by the pub/sub engine include the spatial event model, the spatial subscription model, and the notification model.

Spatial event model

A spatial event is described by a set of properties specified by name and value (NV) pairs. The three mandatory properties for a spatial event are the following:

1. The *object identifier* (*oid*) is a unique identity indicating the owner of the location. The object is a person, a device, or anything that can be located.
2. The *timestamp* is the time at which the object is positioned.
3. The *location* is the geographical location specified by a predefined spatial reference system or a textual description of the location that could be translated into a geographical location via geocoding.

These three properties describe the most important three dimensions that pertain to a spatial event: *who, when,* and *where*. Other optional properties could also be introduced to facilitate the processing of spatial events. These could include *uncertainty*, which describes the uncertainty of the location detected, and a *location provider identifier*, which provides the location information.

Besides the predefined mandatory and optional names for properties, other properties could be attached to the spatial event to describe domain or application-specific information. Usually this information is intended for the subscription application and is not processed by the pub/sub engine.

Spatial subscription model

Spatial subscriptions are used by subscribers to express their interests on spatial events. In the spatial pub/sub system, a spatial subscription is defined as a tuple (*SP*, *ToS*), where *SP* is a *spatial predicate* defined upon the location of objects and *ToS* is the *type of service*, discussed later. The semantic for a spatial subscription is this: *A notification (based on the notification model) is sent to a subscriber when an incoming spatial event (based on the spatial event model) meets SP and ToS requirements*. Currently, two kinds of SPs are supported in the spatial pub/sub system:

1. The *within* predicate has the syntax (oid_1, oid_2, $\cdots$, oid_n) *within* ($zone_1$, $zone_2$, $\cdots$, $zone_n$). The predicate is true if and only if one of the locations of the mobile user (oid_i) is within one of the zones ($zone_j$). A zone is used to associate a named label with an interested region so that not everyone or every application has to define the query polygons for the same thing. A subscription using the *within* predicate can be used to support services such as "Please send me an e-coupon while I am near the ABC shopping mall."
2. The *distance* predicate has the syntax (*oid*) *distance* (D, oid_1, oid_2, $\cdots$, oid_n). The predicate is true if and only

if the distance between *oid* and oid_i $(1 \cdot i \cdot n)$ is less than D. A subscription using a distance trigger can support services such as a "mobile buddy list," which alerts a user when anyone on his predefined buddy list is nearby.

These two SPs can help meet many location-aware application requirements, and we are investigating other SPs for constructing more complex location-aware applications. Traditional pub/sub systems are based on the *publish–match–notify* operational cycle. Therefore, a spatial event is always sent to subscribers when a spatial subscription is satisfied by the event. In the context of location-aware applications, however, the event filter mechanism could sometimes confuse the user. For example, a user defines a *within* subscription, "(oid_1) *within* ($zone_1$)", where $zone_1$ is the predefined area of a shopping mall. While the user (oid_1) is entering the zone ($zone_1$), he receives a promotion message. The *within* subscription is always evaluated to be true when the user stays in the shopping mall, so he continuously receives the promotion message. Obviously, in this case, only one promotion message makes sense to both the user and the promotion provider. A one-time semantic of the subscription specified by the ToS is introduced to guarantee that the user receives only one promotion message. While the ToS is set to once, the one-time semantic of the subscription is applied by the spatial matching engine.

Notification model

Notification takes the same NV pairs format as the event model. It includes two parts:

1. The original event part copied from the event information.
2. The subscription-specific part derived from the process of matching subscriptions with events.

Client-side event processing approach

Owing to the intrinsic constraints of a mobile network, the bandwidth of wireless communication is limited. How to use the available bandwidth in an economic and efficient way is a challenge for the system designer. When spatial event matching is handled in the central pub/sub server, intelligent devices should continuously publish their own location to the server. This consumes bandwidth and restricts the concurrency of the server. A novel client-side event processing approach is presented to solve this problem. This approach takes full advantage of resource capabilities such as computing, storage, and positioning in intelligent mobile devices. Its workflow is as follows:

1. The spatial pub/sub server dispatches within subscriptions to a related intelligent device.
2. The intelligent device obtains its location from the embedded positioning module and performs spatial matching, instead of reporting the location to the pub/sub server.
3. If the spatial subscription is satisfied—for example, if a user is entering a predefined zone—a notification is sent to the pub/sub server. Otherwise, the location is discarded.

By leveraging the computing resources of intelligent devices, the proposed novel event processing approach not only relieves server-side workload, but reduces communication times and saves wireless communication bandwidth, thus enabling the system to support more concurrent mobile users with less cost. Detailed performance results can be found in [8].

Components

As shown in Figure 4, the spatial pub/sub engine is highly modular, with clearly defined interfaces. The spatial pub/sub manager manages the subscription/publishing of an event and exposes the Java Messaging Service (JMS) [9] interface. Message selectors from JMS are extended to support a spatial subscription whose syntax is based on the subscription model. The spatial matching engine takes charge of filtering spatial events. To achieve high performance in spatial matching, the engine uses a spatial index technique to accelerate the matching process. For the *within* predicate, an R-tree is employed to index predefined zones on the basis of their minimum bounding rectangle and to transform the subscription evaluation to R-tree searching. Each zone maintains a hash table to record the list of interested users. When a mobile user's location is within a zone and the mobile user is in the hash table of the zone, a notification is sent out. When the distance evaluation involves more than one mobile user, the location cache is provided to store the latest location data for those users, and an object trigger graph mechanism is employed to handle the case. The spatial match engine reuses the *trigger handler* module from the CAMEL project [5]. The detailed algorithms and a performance comparison can be found in [5]. The results show that the matching engine has good performance and scalability.

The location agent controller and mobile location agent work together to implement the proposed client-side event-processing approach and relieve the workload of the spatial matching engine. The controller manages all mobile location agents running on intelligent devices. It provides an authentication mechanism for these agents, sends related *within* subscriptions to them, receives the matching spatial events from them, and forwards those

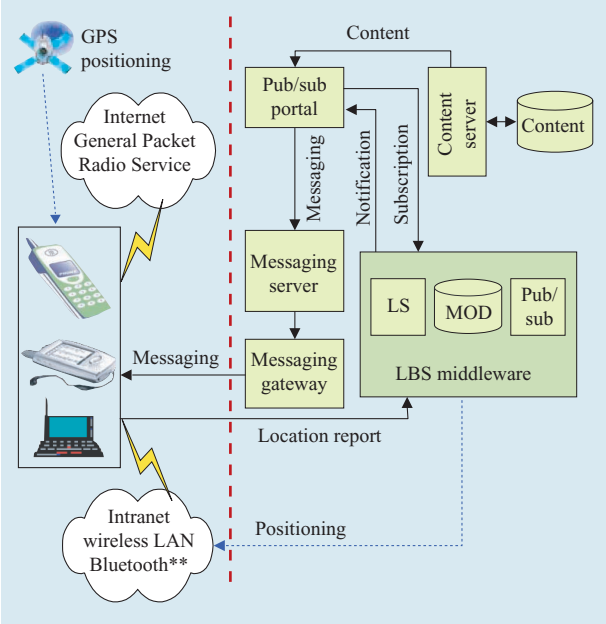

Figure 5

Eagle system architecture.

events to the spatial pub/sub manager. The interface between the controller and the agent is based on XML. The mobile location agent focuses on obtaining its location information from an embedded positioning module, such as GPS, and handling *within* predicates sent by the controller. It evaluates spatial predicates on the basis of the latest mobile device location. When a spatial predicate is evaluated to be true, the matching spatial event is sent to the controller. When the mobile user involves *distance* predicates, the location agent controller periodically queries the mobile location agent for the mobile user's location. When the controller receives the location information back from the agent, it passes it to the spatial matching engine.

The zone definition engine (ZDE) is used by the system administrator and/or end user to define well-known zones of interest (ZOIs) or user-specific ZOIs. A ZOI can be a polygon (including rectangle) or a circle. A ZOI associates a named label with a commonly referenced location such as a shopping mall, so that each individual user does not have to define a query rectangle for the same thing. For example, in Beijing, a place such as Xidan (a large shopping area with a number of giant malls) can be predefined in the system as a polygon on the basis of its geographic coordination (latitude, longitude). Similarly, users can each predefine their homes according to their geographic locations. Predefined ZOIs can be assigned a symbolic name in the system, such as *SYSTEM.Xidan* and *Mike.HOME*, where the prefix *SYSTEM* indicates a system-defined ZOI, and others, preceded by a name, are ZOIs defined by users. The zone definition engine exposes ZDE APIs for client applications and for the spatial pub/sub manager to manipulate zones.

The privacy manager provides a privacy API to handle privacy issues and allows users to control who and what applications can access their location. The discussion of privacy protection is beyond the range of this paper, but is discussed in [10, 11].

Demo prototype—Eagle

Eagle—a proof-of-concept system based on LBS middleware—was built at the IBM China Research Laboratory (CRL) to demonstrate how the LBS middleware could be used to build innovative location-aware applications. The architecture of the system is depicted in **Figure 5**. The subscription is submitted from the pub/sub portal by the CRL host for a guest, Mike. In the portal, some ZOIs can be defined for a specific mobile user. The ZOIs could be the airport, the CRL building, and the reception and conference rooms in the building. The subscription is handled by the pub/sub engine in the LBS middleware. If Mike enters a ZOI, the application is notified with related information (the zone entered, location of user, etc.). Mike's positions are reported to the location server at a predefined interval or whenever his location changes.

In the system, three types of locating methods are used:

- *GPS* is used as the outdoor locating mechanism, and a piece of Java code running on a Compaq iPAQ** PDA periodically reports the location to the location server in the format of (*user ID, longitude, latitude, timestamp*).
- In the CRL building, the location is reported when a user enters a new cell, managed by access points of the wireless LAN.
- While being discovered by some stationary Bluetooth** devices scattered in the building, the device reports their locations to the location server. These Bluetooth devices are placed in the reception area, conference rooms, classrooms, etc.

Upon receiving the location report from mobile devices, the location server publishes the location to the spatial pub/sub engine, which compares it with a list of existing subscriptions. If there is a match, the portal receives the notification (for example, *Mike is now at the airport*) and retrieves related contents from the content server. The content could be a static text message, such as a welcome or an introduction, or it could be a message with dynamic content retrieved from a Web service when the user's location was input. Such content might be a GIS map service or a routing service. The content is forwarded

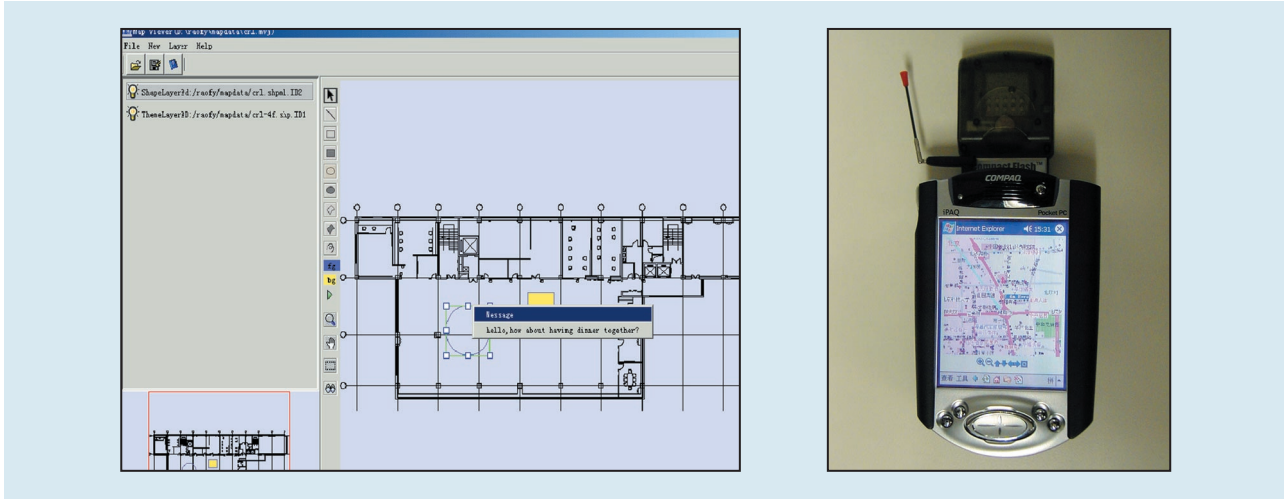

Figure 6

The publish interface and a map of Beijing in iPAQ.

to a messaging server using a Web service interface. The message is delivered to the J2ME messaging client in the PDA through the messaging gateway, which distributes the message to the general packet radio service (GPRS) network or wireless LAN. The messaging client presents the user with the location-dependent content. If applicable, information in the content can be accessed via a URL. **Figure 6** illustrates the publish interface for the IBM China Research Laboratory buildings and a map of Beijing delivered by the system to an iPAQ.

Related work

Considerable effort is being made and considerable resources are being devoted to location-aware computing, both in academia and industry. To our knowledge, however, no comprehensive framework for building location-aware services like the LORE model has been proposed. Several separate efforts relate to partial components or domains within the LORE model. They are discussed below.

Much of the research has focused on developing a services architecture for location-based applications. Building on active badge location technology, researchers at Olivetti proposed and developed the architecture for a distributed location service [12]. The Olivetti system is tied to the specific positioning technology.

Leonhardt presents a global general location service to support location awareness in open distributed systems [13]. He emphasizes the importance of integrating various location techniques and depicts a hierarchical, semisymbolic location model and uses policy-based access control to protect location privacy. Although his effort influences our location model, it is short of abundant auxiliary services to simplify LBS development.

The work of Pfeifer and Popescu-Zeletin [14] has many aspects in common with our work on the location server. They introduce a modular location-aware service and an application platform. The platform provides modular, unified access to various services, which are commonly used by multiple applications. Its prototype, however, is based on CORBA**, and less effort has been directed at privacy control. Privacy is considered to be the critical issue in LBS.

Narayanan presents the idea of logical location contexts which provide enhanced privacy in location-aware mobile computing [10]. His work has helped us to build an advanced privacy control mechanism.

MOD research addresses the issues of storing and processing continuously moving objects. These issues can arise in a wide range of applications, including traffic control and monitoring, transportation and supply chain management, digital battlefields, and mobile e-commerce [15]. The pioneer work in MOD was done by Wolfson at the University of Illinois at Chicago. In the DOMINO prototype [16, 17], the moving object spatio-temporal (MOST) data model [18] was proposed, as was the Future Temporal Logic query language for modeling and querying of moving objects. CAMEL shares the same goal of managing the locations of moving objects and managing the historical locations of moving objects for further use in data mining.

The challenge in MOD is to support dynamic, continuously evolving data and moving queries. As a spatio-temporal database, MOD manages data with spatial

and temporal dimensions. To improve query performance, indexing of moving objects is needed. Several methods of indexing moving objects [19–23] based on R-trees have been put forward in the literature. Some work has also been done to evaluate different indexing methods. Myllymaki and Kaufman [24] investigated the performance and scalability of three main-memory-based spatial indexing methods under dynamic update and query loads. Two data models were investigated on indexing moving objects:

1. *Continuous model.* In the continuous model, moving objects are modeled as points that start moving from a specific location with a constant velocity vector. A MOST data model is used. Some indexing methods—such as time-parameterized R-tree (TPR-tree) [21] and indexing scheme [25]—were developed on the basis of this model.
2. *Discrete model*. In this model, only the locations of a moving object are stored in a database as (*location*, *timestamp*) tuples. Pfoser et al. [20] further developed the model to describe trajectory segments and, to index the historical trajectory of moving objects, proposed two access methods based on the model: spatial-temporal R-tree (STR-tree) and trajectory-bundle tree (TB-tree).

The continuous model implicitly requires that moving objects report both their location and velocity to the location management system. This model is feasible in areas such as vehicle tracking and digital battlefields, where advanced devices equipped with GPS receivers can report the required data to the system. However, in location-based services which serve mobile users, a large portion of potential customers have only cell phones, which have no self-positioning capability and require assistance from the wireless network to determine their locations. Thus, in real life, it is difficult to directly determine the velocities of most mobile users. For this reason, CAMEL employs a discrete location model instead of a continuous model, and it receives only location information with timestamps from a tracking server or from the device itself. The inherent differences between the two models suggest different ways to process queries and to index locations.

Gryphon [26] and Siena [27] are content-based pub/sub systems. Gryphon provides an efficient and scalable filtering algorithm to handle event matching. Siena provides an expressive subscription language for subscribers to select events of interest. Both of them support primitive data types, while our spatial pub/sub system can handle complicated spatial data types. Podnar et al. [28] presented an architecture to deliver content to mobile users on the basis of the pub/sub paradigm. Their pub/sub system can work together with location management to deliver location-aware messages to mobile users. However, the spatial pub/sub system proposed in this paper focuses on spatial-related information matching and performance improvement.

An event-specification language that can be used to express spatial events was presented by Bauer and Rothermel in [29]. Their semantics of basic spatial events is the same as ours, but they paid more attention to event definition and event composition, while we made a greater effort on spatial subscription and spatial event matching.

Work has been done at the University of Cambridge to investigate services based on registration of interest in user locations and proximities and notifying clients when changes occur [30]. That system architecture, called CALAIS, is based on distributed events technology and a dynamically modifiable R-tree index. Fed with a stream of location events, it was used to monitor locations and proximities. CALAIS is suitable for the support of context-aware applications operating within a typical indoor office domain, while our spatial pub/sub system is more suitable for the outdoor environment because it leverages intelligent devices to provide high-performance spatial event processing. A scalable location-aware system, Rover, developed at the University of Maryland by Banerjee et al. [31], provides a function similar to that of our Eagle demo. Rover focuses on achieving system scalability to very large client sets by introducing an action-model-concurrent software architecture, and our Eagle demo targets high-performance spatial pub/sub mechanisms.

Conclusion and future work

This paper presents the LORE model, including domains of location operation semantic, privacy and security, management, and a location-aware agent. Based on the LORE model, an infrastructure, LBS middleware, was built to support the rich sets of location-aware applications. Three key components of LBS middleware—the location server, the moving object database, and the spatial pub/sub engine—implement the domain components of the LORE model, enable innovative location-aware applications, and make possible new user experiences. A proof-of-concept system, Eagle, is demonstrated to show how a location-aware service can be built on such an infrastructure.

Some open issues in the LORE model still have to be addressed and integrated into the LBS middleware. Support for the fusion component must be investigated in order to achieve more accurate location information from multiple location sources. The privacy model and privacy provisioning mechanism should be developed to protect users' privacy while they enjoy new services, especially services based on intelligent notification. The LBS middleware requires enhanced and integrated management

tools to facilitate the deployment and management of location-aware services and to align with other information technology applications in enterprise or computing environments.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademarks or registered trademark of the GSM Association, Sun Microsystems, Inc., Telefonaktiebolaget LM Ericsson, Cisco Systems, Inc., Bluetooth Special Interest Group, Hewlett-Packard Development Company, L.P., or Object Management Group, Inc. in the United States, other countries, or both.

References

1. C. A. Patterson, R. R. Muntz, and C. M. Pancake, "Challenges in Location-Aware Computing," *IEEE Pervasive Computing* **1,** No. 2, 80–89 (April–June 2003).
2. Location Inter-Operability Forum (LIF); see *http://www.locationforum.org/*.
3. J. Myllymaki and S. Edlund, "Location Aggregation from Multiple Sources," *Proceedings of the 3rd International Conference on Mobile Data Management (MDM)*, January 2002, pp. 131–138.
4. WAP Location Protocol; see *http://www.wapforum.org/*.
5. Y. Chen, F. Rao, X. Yu, and D. Liu, "CAMEL: A Moving Object Database Approach for Intelligent Location Aware Services," *Proceedings of the International Conference on Mobile Data Management*, January 2003, pp. 331–334.
6. Community Development of Java Specifications; see *http://jcp.org/en/jsr/detail?id=179*.
7. D. Kwon, S. Lee, and S. Lee, "Indexing the Current Positions of Moving Objects Using the Lazy Update R-Tree," *Proceedings of the 3rd IEEE International Conference on Mobile Data Management*, January 2002, pp. 113–120.
8. X. Chen, Y. Chen, and F. Rao, "An Efficient Spatial Publish Subscribe System for Intelligent Location-Based Services," *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS '03)*, June 2003; see *http://www.eecg.toronto.edu/debs03/papers/chen_etal_debs03.pdf*.
9. Java Messaging Service; see *http://java.sun.com/products/jms/*.
10. A. K. Narayanan, "Realms and States: A Framework for Location Aware Mobile Computing," *Proceedings of the 1st International Workshop on Mobile Commerce*, July 2001, pp. 48–54.
11. E. Snekkenes, "Concepts for Personal Location Privacy Policies," *Proceedings of the ACM Conference on Electronic Commerce (EC '01)*, October 2001, pp. 48–57; see *http://www2.hig.no/~einars/einar_publications/papers/ACM_EC01_13_09_2001.pdf*.
12. A. Harter and A. Hopper, "A Distributed Location System for the Active Office," *IEEE Network* **8,** No. 1, 62–70 (January/February 1994).
13. U. Leonhardt, "Supporting Location-Awareness in Open Distributed Systems," Ph.D. thesis, Imperial College of Science, Technology, and Medicine, University of London, England, 1998.
14. T. Pfeifer and R. Popescu-Zeletin, "A Modular Location-Aware Service and Application Platform," *Proceedings of the 4th IEEE Symposium on Computers and Communications*, July 1999, pp. 137–148.
15. O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving Objects Databases: Issues and Solutions," *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, April 1998, pp. 111–122.
16. O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha, "Updating and Querying Databases that Track Mobile Units," *Distributed & Parallel Databases* **7,** No. 3, 257–287 (1999).
17. O. Wolfson, A. P. Sistla, B. Xu, J. Zhou, and S. Chamberlain, "DOMINO: Database for Moving Objects Tracking," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, June 1999, pp. 547–549.
18. A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," *Proceedings of the 13th IEEE International Conference on Data Engineering*, April 1997, pp. 422–432.
19. P. K. Agarwal, L. Arge, and J. Erickson, "Indexing Moving Points," *Proceedings of the 19th Annual ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems*, May 2000, pp. 175–186.
20. D. Pfoser, Y. Theodoridis, and C. S. Jensen, "Novel Approaches in Query Processing for Moving Object Trajectories," *Proceedings of the 26th International Conference on Very Large Data Bases*, September 2000, pp. 395–406.
21. S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the Positions of Continuously Moving Objects," *Proceedings of the ACM SIGMOD/PODS Conference*, May 2000, pp. 331–342.
22. G. Kollios, V. J. Tsotras, D. Gunopulos, A. Delis, and M. Hadjieleftherious, "Indexing Animated Objects Using Spatiotemporal Access Methods," *TimeCenter Technical Report TR-54*, January 25, 2001; see *http://www.cs.auc.dk/research/DP/tdb/TimeCenter/TimeCenterPublications/TR-54.pdf*.
23. S. Saltenis and C. S. Jensen, "Indexing of Moving Objects for Location-Based Services," *Proceedings of the IEEE International Conference on Data Engineering*, February 2002, pp. 463–472.
24. J. Myllymaki and J. Kaufman, "High-Performance Spatial Indexing for Location-Based Services," *Proceedings of the 12th International World Wide Web Conference*, May 2003, pp. 112–117; see *http://www2003.org/cdrom/papers/refereed/p612/p612-myllymaki.html*.
25. G. Kollios, D. Gunopulos, and V. J. Tsotras, "On Indexing Mobile Objects," *Proceedings of the 18th ACM Symposium on Principles of Database Systems*, May 1999, pp. 261–272.
26. M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, "Matching Events in a Content-Based Subscription System," *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC '99)*, May 1999, pp. 53–61.
27. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service," *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC '2000)*, July 2000, pp. 219–227.
28. I. Podnar, M. Hauswirth, and M. Jazayeri, "Mobile Push: Delivering Content to Mobile Users," *Proceedings of the 22nd International Conference on Distributed Computing System Workshops (ICDCSW '02)*, July 2002, pp. 563–570.
29. M. Bauer and K. Rothermel, "Towards the Observation of Spatial Events in Distributed Location-Aware Systems," *Proceedings of the 22nd International Conference on Distributed Computing System Workshops (ICDCSW '02)*, July 2002, pp. 581–582.
30. G. Nelson, "Context-Aware and Location Systems," Ph.D. thesis, Clare College, University of Cambridge Computer Laboratory, January 1998; see *www.sigmobile.org/phd/1998/theses/nelson.pdf*.

31. S. Banerjee, S. Agarwal, K. Kamel, A. Kochut, C. Kommareddy, T. Nadeem, P. Thakkar, B. Trinh, A. M. Youssef, M. Youssef, R. L. Larsen, A. U. Shankar, and A. K. Agrawala, "Rover: Scalable Location-Aware Computing," *IEEE Computer* **35,** No. 10, 46–53 (October 2002).

Received October 17, 2003; accepted for publication January 29, 2004; Internet publication September 17, 2004

Ying Chen *IBM Research Division, IBM China Research Laboratory, 4/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (yingch@cn.ibm.com).* Dr. Chen received B.S. and Ph.D. degrees in computer science from Southeast University, China, in 1994 and 1999, respectively. He subsequently joined the IBM China Research Laboratory and is currently a Research Staff Member and manager. He led location-based services and spatial information processing projects and contributed to IBM products. He published more than 20 technical papers and filed several patents. Dr. Chen's research interests include Grid computing, data management, and service-oriented architecture. He currently leads the operating environment research team and works on Web services, Grid computing, and their application in the telecommunications industry.

Xiao Yan Chen *IBM Research Division, IBM China Research Laboratory, 4/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (xiaoyanc@cn.ibm.com).* Ms. Chen is a Research Staff Member in the On-Demand Technologies Department. She received B.S. and M.S. degrees from Peking University in 1988 and 1991, respectively, joining IBM in 1999. She contributed to the design and development of location-based services middleware for the IBM WebSphere Everyplace Suite. Ms. Chen has published several papers in pervasive computing and holds three patents. She is currently responsible for value net management environment to enhance the Operating Supporting System/Business Supporting System.

Fang Yan Rao *IBM Research Division, IBM China Research Laboratory, 4/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (raofy@cn.ibm.com).* Ms. Rao is a Research Staff Member. She received a B.S. degree in computer science from Xi'an Jiaotong University in 1993, and an M.S. degree in network and distributed computing from Xi'an Jiaotong University in 2000. She subsequently joined the IBM China Research Laboratory, where she has worked in the location-based services and spatial data processing areas. She is currently working on technologies related to Grid and Web services. Ms. Rao has received IBM Invention Achievement Awards for her work, including spatial index optimization.

Xiu Lan Yu *IBM Research Division, IBM China Research Laboratory, 4/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (yuxl@cn.ibm.com).* Dr. Yu received B.S., M.E., and Ph.D. degrees from the Harbin Institute of Technology in 1994, 1996, and 1999, respectively. She was a postdoctoral fellow in Chinese Academic Sciences, joining IBM in 2001. While at IBM she has contributed to the research of location-based services, with a focus on component design, simulation, and modeling, as well as scalability. Dr. Yu is currently responsible for research on the service operating environment.

Ying Li *IBM Research Division, IBM China Research Laboratory, 4/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (lying@cn.ibm.com).* Dr. Li received B.S., M.E., and Ph.D.

degrees from the Northwest Polytechnic University in 1996, 1998, and 2001, respectively. She has been a Research Staff Member since joining IBM in 2001. While at IBM she has contributed to the research and development of location-based services, spatial business intelligence, and Web/Grid services technologies. She has authored ten technical papers and holds one patent on Web services. Dr. Li is currently responsible for researching resilient Web/Grid services provisioning and management for infrastructure for IBM on-demand e-business.

Dong Liu *IBM Research Division, IBM China Research Laboratory, 4/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (liudong@cn.ibm.com).* Dr. Liu received B.S., M.S., and Ph.D. degrees from Peking University in 1990, 1993, and 1996, respectively. He has been a Research Staff Member and manager since joining IBM in 1996. His research interests include database, pervasive computing, and business process modeling. He has contributed to the spatial index tuner, location-based services, and Session Initiation protocol which are now part of IBM DB2 and WebSphere. He has authored more than 20 technical papers and filed several patents. Dr. Liu currently leads the Business Integration Department and works on business process modeling and optimization.