Data Management in Location-Dependent Information Services

Location-dependent information services have great promise for mobile and pervasive computing environments. They can provide local and nonlocal news, weather, and traffic reports as well as directory services. Before they can be implemented on a large scale, however, several research issues must be addressed.

Dik Lun Lee, Jianliang Xu, and Baihua Zheng
Hong Kong University of Science and Technology

Wang-Chien Lee
Pennsylvania State University

Location-dependent information services (LDISs) are an important class of context-aware applications. They answer location-related queries, where a location is either explicit or implied. These services emerged from advances and convergence in high-speed wireless networks, personal portable devices, and location-identification techniques. With a variety of promising applications, such as local information access (traffic reports, news, navigation maps, and so on) and nearest-neighbor queries (such as finding the nearest restaurant), LDISs will soon become an integral part of our daily lives.

Although LDISs exist in traditional computing environments (Guides@Yahoo, for example), their greatest potential is in a mobile-pervasive computing environment, where users enjoy unrestricted mobility and ubiquitous information access.

In this article, we discuss location-dependent information access in a mobile-pervasive environment, in particular in a cellular mobile system, and present new research issues arising from on-demand access, broadcast, and data caching.

Preliminaries

Managing data in an LDIS faces several challenges:

- *Mobile environment constraints*. Mobile-pervasive environments offer scarce bandwidth, low-quality communication, frequent network disconnections, and limited local resources, complicating the provision of location-dependent information to mobile users.
- *Spatial data*. In an LDIS, answers to user queries can vary with location. That is, query results depend on a query's spatial properties. For a location-bound query, the query result must be both relevant to the query and valid for the bound location.
- *User movement*. Because mobile users change locations, some tasks—such as query scheduling and cache management—are particularly tough in an LDIS.

These challenges have opened many new research problems in LDIS data management. Although many traditional data management techniques are applicable to the implementation of an LDIS, we must reexamine and recreate them to address these issues. Many studies address data management for mobile computing, such as cache management and transaction management.[1] However, most existing studies target general data services instead of exploring the special properties of LDISs.

Location models

Locations in an LDIS must be specified explicitly or implicitly before a client can access information. A location model depends on the system's underlying location identification technique. Two models for representing locations exist:

1536-1268/02/$17.00 © 2002 IEEE

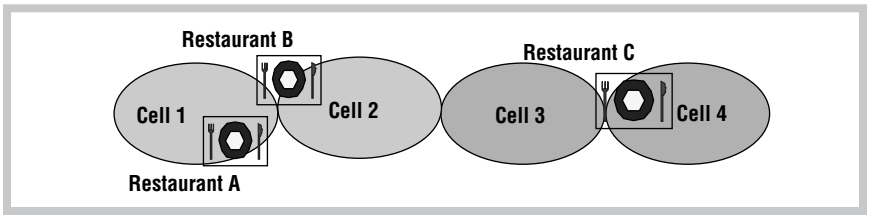


Figure 1. Valid scope for a four-cell system with a wireless cell-based location model.

- *Geometric model.* The system specifies a location as an n-dimensional coordinate (typically $n = 2$ or 3)—for example, the latitude-longitude pair returned by the Global Positioning System—or a set of coordinates defining an area's bounding geometric shape (such as a polygon). The system can compute regular geometric shapes from concise representations (for example, a circle can be represented by the center and radius). The geometric model's main advantage is its compatibility across heterogeneous systems. However, it can be very costly and complex in terms of the volume of data involved and the need to map the geometric representation to a semantic level suitable for the applications.
- *Symbolic model.* Logical, real-world entities describe the location space. Entities can be buildings, streets, cities, or system-defined elements such as wireless cells, and are uniquely identifiable by a hierarchical naming system. The symbolic model typically has coarser location granularity than the geometric model because it stresses the representation of relationships between logical entities rather than their precise coordinates. At a semantic level, it is more suitable for LDIS applications. Also, being discrete and well-structured, symbolic location information is easier to manage. Converting locations among heterogeneous systems, however, is difficult.

The geometric and symbolic location models have different overheads in identifying locations and represent location information at different levels of precision. The appropriate location model depends on the application.

An LDIS often needs location information expressed in both models. On one hand, the system must map a geometric location into a symbolic model before using it to query objects expressed in the symbolic model. For example, an LDIS maps a geometric location obtained from GPS to a shopping mall, and then the client can query all the mall restaurants. On the other hand, many operations such as finding the distance between two objects (for example, between a shopping mall and a train station) must be computed in geometric coordinates.

Query types

Different query types require different indexing and query-processing strategies. We can roughly classify location-dependent queries into two dimensions. (Ayse Y. Seydim and her colleagues provide a formal classification.[2])

Local vs. nonlocal queries. Local queries are bound to a user's current location (for example, "list the local weather" and "find the nearest restaurant"). Nonlocal queries are bound to locations other than the user's current location ("find the weather in New York City" and "find the nearest station to the White House," for example). A system can resolve local queries based on the user's local context, whereas for nonlocal queries, it must first identify the contexts in which it will evaluate the queries.

Simple vs. general queries. Simple queries place simple equality conditions on the underlying data. For example, the query, "download the local traffic report," retrieves the traffic report associated with the user's current wireless cell. A query like "find the nearest restaurant" can also be viewed as a simple query because "nearest restaurant" is a functional attribute that finds the nearest restaurant based on the user's location.

General queries involve complex conditions on the underlying data. They can be further divided into *spatially constrained* queries, in which at least one predicate has spatial properties, such as "list the hotels within 10 miles"; and *nonspatially constrained* queries, which only contain predicates without spatial properties (for example, "list the hotels with a room rate below $100"). Common operators in spatially-constrained queries include **intersect**, **contain**, **within**, and **distance**.[2] Techniques for processing general queries are more sophisticated because the data types (scalar versus vector) and operations are different.

Valid scopes

Recall that in an LDIS a query result depends on the location the query is concerned with. For caching and indexing purposes, it is useful for the LDIS to return the result and its *valid scope*—that is, the area or areas within which the query result is valid. Valid scopes are important in developing effective data placement, indexing, cache replacement, and cache invalidation methods.

A query returns a data instance, which is represented as a tuple (query, result). We use *query* to refer to a string encoding the returned data item's name and the condition imposed on it. For example, if hotel A is the nearest hotel within a particular area and the cheapest hotel within a potentially different area, two data instances exist: (**nearest-hotel, A**) and (**cheapest-hotel, A**). For the general query, "return the cheapest hotel that is next to a cinema," the data instance is represented as (**cheapest-hotel-next-to-a-cinema, C**). Note that by this definition, each data instance has only one valid scope.

In the geometric location model, a valid scope often takes the shape of a polygon in a 2D space. The symbolic location model represents a valid scope as a set of logical zone IDs. Figure 1 shows a four-cell system with a wireless cell-based location model. Suppose that the nearby restau-

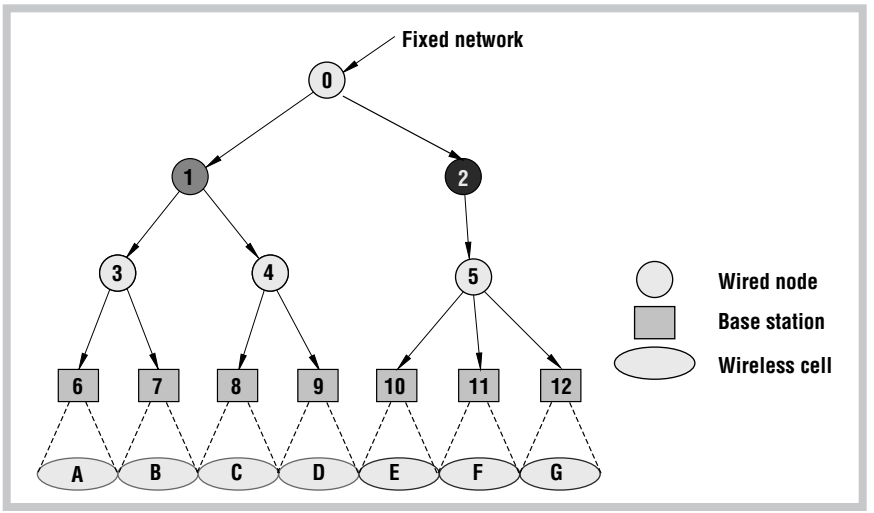


Figure 2. The cellular architecture for LDIS systems. Wireless cells are serviced by base stations and connected by wired nodes through wireless links.

rants are A and B for cells 1 and 2 and C for cells 3 and 4. Then, for the nearby restaurant query, the valid scope of the data instance (nearby-restaurant, {A, B}) is {1, 2}; the valid scope of the data instance (nearby-restaurant, {C}) is {3, 4}.

Research issues

In a wireless system, users can access data on demand, by broadcast, or by a combination of the two. We can apply data caching to each mode of access to improve system performance. When adopting a data access method for an LDIS, developers should consider resource availability, cost, and performance. The choice of method is also limited by what is available from the underlying wireless communication infrastructure.

The different data access methods can affect LDIS implementation significantly, and each method brings with it various technical issues. We discuss these issues under a cellular architecture, which is illustrated in Figure 2.[1] The system's geographical coverage area is partitioned into wireless cells. Each wireless cell is serviced by a *base station*, which is a leaf node in the fixed network. A mobile client connects to a corresponding base station over wireless links to access relevant information from the fixed server.

On-demand access

In on-demand access, a mobile client submits a request, consisting of a query and the query's associated location, to the server. The server locates the appropriate data and returns it to the mobile client either individually or by multicast. As the most fundamental data access mechanism, on-demand access can be used for all types of queries. We have identified four main issues in on-demand access: data placement, data replication, query scheduling, and indexing.

Data placement. The first issue developers must face when dealing with location-dependent data is where to place them in the system architecture. In a centralized solution, all data go into a central database (for example, the root in Figure 2). Obviously, this scheme is unattractive in terms of performance and robustness because the database can become a bottleneck or a single point of failure.

Alternatively, we can use a distributed placement solution. To do this, we store the data in selected wired nodes, which form a hierarchy. We then place a data instance at the least common ancestor node for the data instance's valid scope. For example, in Figure 2, we would store a data instance whose valid scope falls into cells A to D at wired node 1.

The two placement solutions represent valid scopes and process queries differently. A centralized solution cannot scale to a large system because it is impractical for the central database to store the valid scopes of all data instances and serve queries from all cells. In a distributed solution, the valid scopes are distributed across many nodes, which eases the storage demand on a single node. Furthermore, data placement partly implies the valid scopes. For example, in Figure 2, if we use a cell-based symbolic location model, a data instance placed in a base station implies that it is valid for that cell and thus no explicit valid scope need be stored. A data instance stored at wired node 1 implies that the instance is valid in some or all of cells A to D but invalid in cells E to G. Thus, we can use a more compact representation (for example, by using a four-bit vector instead of a seven-bit vector to identify the valid scopes). We can realize similar savings when using a geometric model. In summary, a distributed method results in a more uniform workload and more efficient querying; however, managing valid scopes is more complex.

The data placement solution used also affects query processing. In a centralized solution, the centralized data server processes all types of queries, although not always efficiently.

Although a distributed solution simplifies processing of simple queries, it complicates the processing of general queries. To process a local simple query, the system searches the servers from the corresponding base station up the database hierarchy until it finds the relevant instance. The length of the path from the query point to the solution is typically quite short. Moreover, the smaller candidate set and less resource contention make query latencies in distributed solutions shorter than in centralized solutions. The simplest way to process a nonlocal simple query such as "find the local news for New York City" is to redirect it to the specified remote cell(s) and then follow the same process used for a local query. On the other hand, a general query such as "find the nearest hotels with a room rate below $100" can incur an

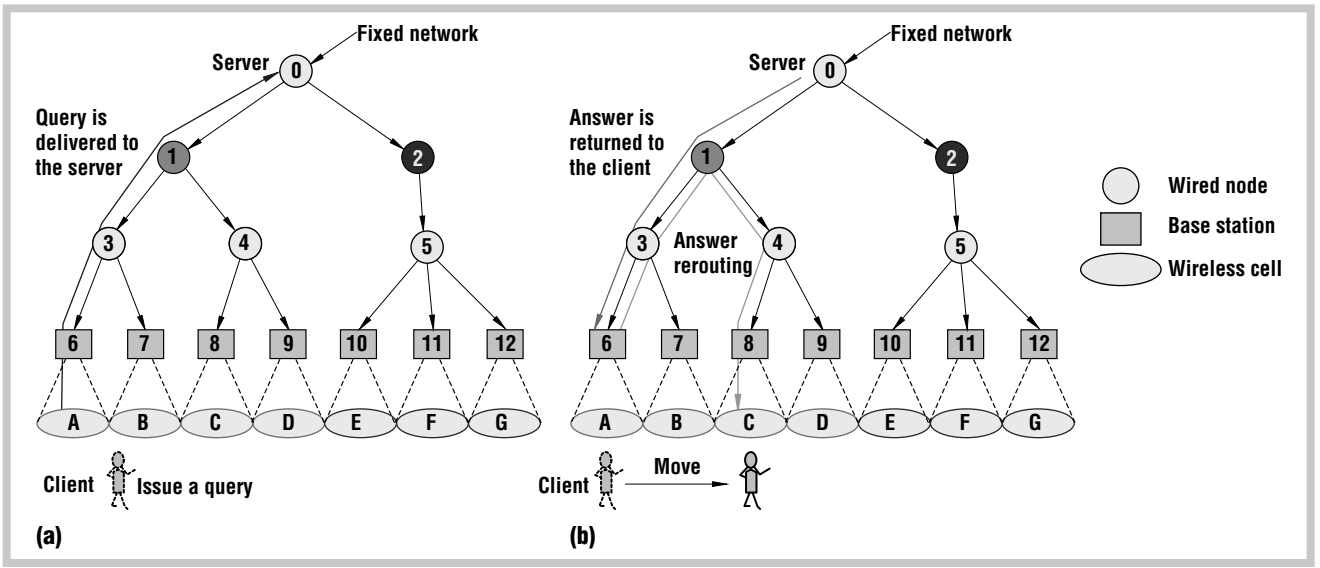


Figure 3. Answer rerouting. A client (a) issues a location-dependent query in cell A, but has (b) already moved to cell C when the answer arrives.

evaluation of the query on many nodes, because, for example, the system might need to process such a query at several distributed nodes and combine the intermediate results returned from the nodes to get the final answer.

Further research should address the following questions: How should a system process a general query? Is it better to fetch the data from neighboring cells and then process the query or dispatch the query to neighboring cells and combine the results? What query optimization techniques will improve the communication cost and response time? What is the granularity of the neighboring cells from which to fetch data or dispatch queries?

Data replication. Developers often use data replication to improve reliability and reduce costs, such as network traffic and response time. Using data replication, the system creates one or more copies of the data, which it places at different locations in the network. A key issue in data replication is placement—that is, what are the replication units, how many replicas should the system create, and where should it put them? Some studies have addressed these issues given network topology and access patterns.[3] To apply these solutions to the LDIS replication problem, we must estimate the access patterns for each data instance at the cell level. This is relatively easy for on-demand access but difficult for broadcast data.

In addition, for location-dependent data, access patterns might be time dependent and periodically repetitive. For instance, people tend to access local weather in the morning and news in the evening. Due to time differences between areas, the access pattern probably slowly migrates from one cell to another. Existing automatic replica migration protocols can adapt to user access behavior changes in distributed systems and Internet environments.[4] We do not yet know, however, whether these protocols would work well for location-dependent data.

Query scheduling. Query scheduling determines query-processing order. In previous work, we sought to improve the average-case queuing delay on the server.[5] User mobility poses a new challenge for query scheduling. Often, a server will return a query result to the cell from which the query was issued after the mobile user has left, as Figure 3 shows.

Although this might not be a big problem for nonlocal queries—the change of user location only affects the path along which the answer is returned—it is an issue for local queries. If the user's location has changed when an answer arrives, the answer might not be valid. The user could reject the invalid answer or require the server to reprocess the query for its current location. Neither case is desirable. In the former, user satisfaction would be poor; in the latter, both network and server performance would suffer. Thus, a server should first schedule the queries that will leave the current answer's valid scope (to avoid reprocessing the query) or the current cell (to avoid rerouting the answer) soonest. Because mobile users can move freely, however, it is often difficult to predict if and when they will leave a certain area. Future work will investigate optimal query-scheduling algorithms for different performance objectives.

Indexing. Data are stored on the servers, which are typically administered by database management systems. Developers often use *disk indexing*—a process in which the server precomputes index information and stores it with the data to allow index searching—to enhance query performance.[6]

For simple queries, we can build an

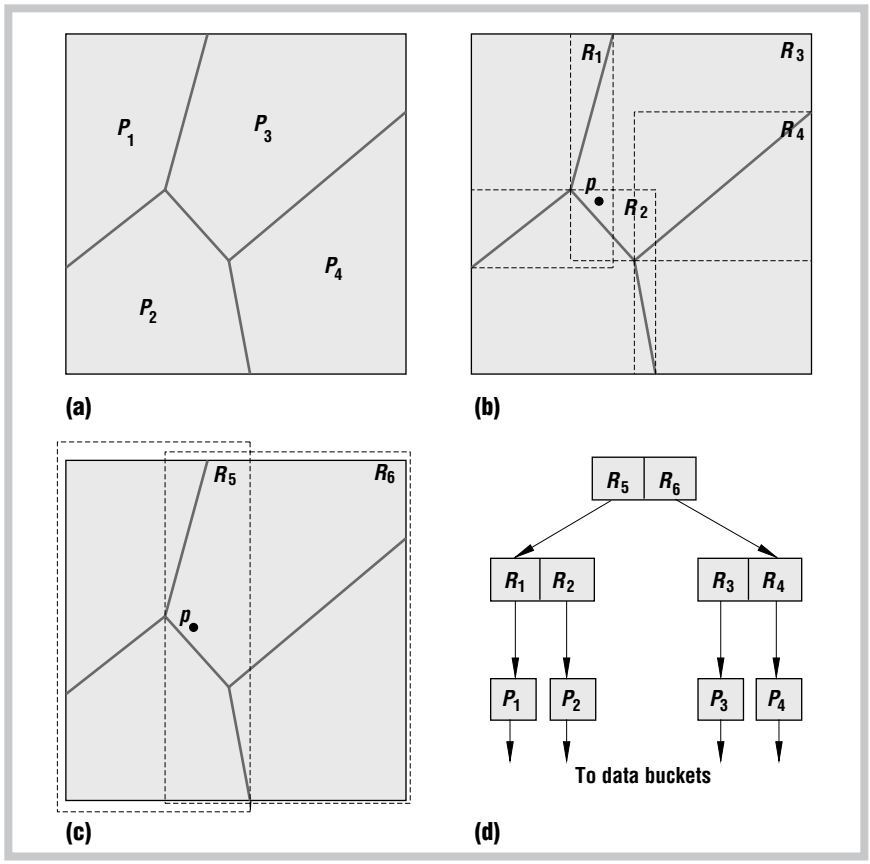


Figure 4. Traditional indexing: (a) valid scopes of data instances for a certain query type, (b) minimal bounding rectangles (MBRs) of the valid scopes, (c) MBRs in the index's internal nodes, and (d) *R**-tree index structure.

index for each query type ("nearest restaurant" is one query type, for example). We abstract the indexing problem as follows: Given the valid scopes of all data instances of a certain query type, how can we index them to allow efficient processing of location-dependent queries? In the symbolic location model, we can construct an index array to map each symbolic area (such as a wireless cell) to a valid data instance in the corresponding area.

In the geometric location model, the problem is more challenging. Traditional spatial index structures often approximate spatial objects using simpler shapes, such as minimal bounding rectangles (MBRs), before the server inserts them in the index.[6] Linking geometric valid scopes renders traditional index structures inefficient, because after approximation, overlapping of MBRs leads to poor search performance.

Figure 4 gives an example of traditional indexing. Figure 4b shows the MBRs R_1, R_2, R_3, and R_4 of the four valid scopes P_1, P_2, P_3, and P_4 in Figure 4a. Figure 4d shows the corresponding R^*-tree index. Given a query point p, as in Figures 4b and 4c, the search first reaches the leaf node R_1 through the root and R_5. Because p is outside P_1, the server rolls the search back to R_2. It is also outside P_2, so it next rolls the search back to R_6. Finally, it obtains the correct answer in node R_3. Even in this simple example, overlapping among sibling MBRs causes many false searches and deteriorates the overall search time. We expect future work to address this issue and to develop sophisticated index structures for point queries.

Overlapping of MBRs is an issue for spatial range queries (for example, "download the traffic report within 10 miles") as well. Future work could investigate efficient index structures supporting various types of location-dependent queries.

Broadcast

In this method, data are broadcast on a public wireless channel. When a mobile client receives a query from its user, it tunes into the broadcast channel and filters out the data according to the query and its associated location. Broadcast lets an arbitrary number of users simultaneously access data.[7] We envisage many LDISs, such as regionwide traffic reports and tourism information, using broadcast to disseminate information to the rapidly increasing mobile population. For general queries, however, broadcast is inefficient because such queries require complex query-processing techniques, such as join of two relations. We thus focus on simple queries and related issues.

Indexing on air. Air indexing is often used to save battery power, which is scarce for mobile clients.[8] In air indexing, the server precomputes index information and interleaves it with the data on the broadcast channel. Consequently, by searching the index, a mobile client can predict the desired data's arrival time. Thus, it can stay in power-saving mode and tune into the broadcast channel only when the requested data arrives. The drawback is that the additional indexing information makes the broadcast cycles longer.

Two aspects distinguish air indexing from disk indexing.

- In air indexing, the index size relates to access latency. Thus, when evaluating an index structure, index size, in addition to search performance, is a big concern.
- Air indexing only allows sequential access, making some traditional indices and search algorithms inefficient.

Consider a nearest-neighbor search. Suppose we have three index entries: the root, R_1, and R_2 (see Figure 5a). Further suppose that given some query point, we can achieve better performance by accessing index entries in the order root, R_2, R_1.

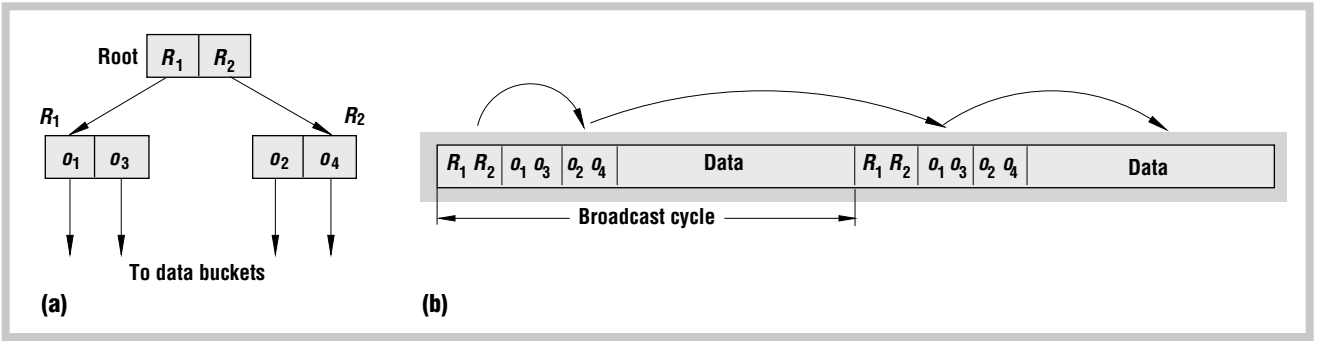


Figure 5. Sequential access in wireless broadcasts: (a) the index contains three entries; (b) the search accesses the index entries sequentially.

However, if we broadcast the index on the wireless channel in the order root, R_1, R_2, the original search order will cause a significant access delay because after the search accesses R_2, it has to wait for the next index broadcast to access R_1 (as Figure 5b illustrates). Ideally, the client would prefetch R_1, but this would require a scheme that balances the amount of prefetched data with power and memory consumption. Therefore, a good index structure for broadcasting location-dependent data should keep the index as small as possible while achieving a good search performance under sequential access.

Several indexing techniques for the wireless broadcast channel exist in the literature. The proposed techniques serve equality-based queries by directing them to right positions using a hashing function[9] or filtering out irrelevant information based on the signature approach.[10] However, we cannot apply these studies to location-dependent query processing that involves spatial queries (such as point queries and range queries) other than equality-based queries. Tomasz Imielinski and his colleagues investigated interleaving the index and the data on the linear wireless channel to optimize tuning time while maintaining a short access latency.[8]

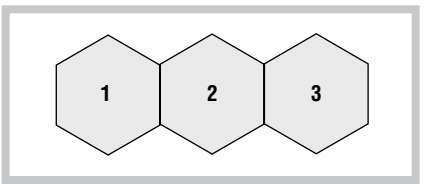


Figure 6. Three cells with possible frequency reuse in cells 1 and 3.

Other work could investigate efficient index structures to support access of location-dependent data on the linear wireless broadcast channel.

Broadcasting strategies. We see two basic broadcasting strategies for providing location-dependent data to mobile users:

- A single wireless channel covers the whole service area. Mobile clients use the broadcast global index to locate desired data.
- The wireless channel is divided into subchannels, and different cells are assigned different subchannels. Each cell broadcasts only the data of interest to its local mobile users, so the amount of data and hence the broadcast index size is smaller. This strategy will have a better access performance if location-dependent queries exhibit high locality.

We use a simple example to illustrate the performance of these two strategies. Suppose that a system consists of three cells, as in Figure 6, and that the available overall bandwidth is 3 kbits per second. Frequency can be reused in cells 1 and 3 because they are not adjacent. In the first strategy, the three cells merge into a larger cell, and the system assigns the new cell a bandwidth of 3 kbps. In the second strategy, the system assigns each cell a bandwidth of 1.5 kbps. There are three data instances, each with 1 kbit. We also assume that the system uses flat broadcast. If the access probabilities for the three instances are the same (that is, 1/3) in all three cells, the expected waiting time is $1/2 \times (1 + 1 + 1)/3 = 0.5$ for the first strategy and $1/2 \times (1 + 1 + 1)/1.5 = 1.0$ for the second strategy.

However, if cells 1, 2, and 3 only access instances 1, 2, and 3, respectively, each cell broadcasts only one instance with the second strategy. The first strategy has the same waiting time (that is, 0.5), whereas the waiting time for the second strategy is $1/2 \times 1/1.5 = 0.33$. In a cellular mobile system, local optimization and frequency interference and reuse complicate this issue. Researchers could pursue further work in this direction.

Data caching

In the data-caching method, data is cached at the mobile client. When the client receives a query, it first searches its cache. If there is a valid copy in the cache, it returns an answer immediately. If not, the client attempts to obtain the data item from the server through on-demand access or broadcast.

Client data caching is particularly important in mobile environments. In addition to improving access latency, data caching can save power due to lower data transmission and improve data availability in case of disconnection.

Location-dependent cache invalidation. Maintaining cache consistency is a major issue in data caching. In general, cached data become obsolete if they are updated

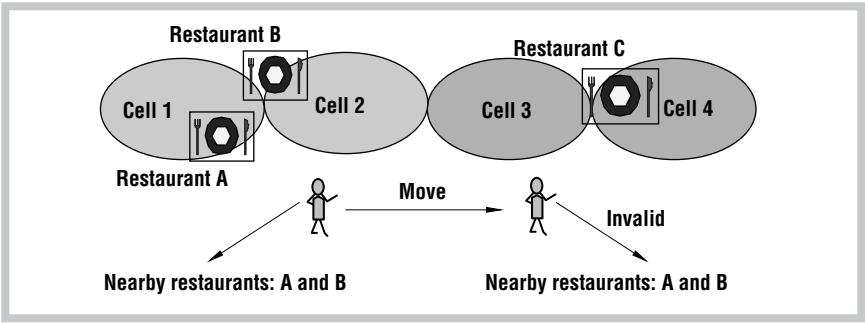


Figure 7. Location-dependent cache invalidation. When the mobile user moves from cell 2 to cell 3, the results from query "find the nearest restaurant" are no longer valid.

on the data server. We call such invalidation *time-dependent cache invalidation*. In LDISs, *location-dependent cache invalidation* is also an issue. In location-dependent cache invalidation, cached data become invalid when a mobile user changes location. For example, a mobile user queries the nearby restaurant at location *A* and caches the answer under "nearby restaurant." Later, the user moves to location *B* and issues the same query. The cached data might or might not be valid at location *B*. In the example in Figure 7, when the user moves from cell 2 to cell 3, the cached result for the query becomes invalid. In addition, data updates on the server can result in out-of-date location-dependent data such as traffic reports. Determining how to effectively perform both time- and location-dependent invalidation is a new area to be explored.

A common way to perform location-dependent cache invalidation is to attach the valid scope to a data instance returned to the client. The client caches the data as well as its valid scope so it can use the cached valid scope for later validity checking.[11] Hence, an important aspect of location-dependent invalidation is identifying and representing the valid scopes, which in turn depend on the location model used. We have proposed different schemes for the symbolic location mode[11] and the geometric location model.[12] However, current work is limited to simple queries.

Cache replacement and prefetching. Because a mobile client has only limited cache space, cache replacement is another issue. In traditional cache management, access probability is considered the most important factor affecting cache performance. A probability-based policy involves replacing the data with the least access probability. In LDISs, there are two additional factors for local queries: data distance and valid scope area.

Data distance refers to the distance between a mobile client's current location and a data instance's valid scope. When a data instance's valid scope is far from the client's current location, the data will have a smaller chance at reusability because it will be some time before the client reenters the valid scope area, and the data is useless before the user reaches the valid scope area. In this respect, we should favor ejecting the "farthest" data during replacement. However, this reasoning is invalid in two cases:

- If the client continues to move away from a location, this location is less likely to be revisited even though it is very close to the client's current location. Thus, a directional data distance would make more sense.
- With random movement patterns, the time it takes the client to traverse a distance is not always directly proportional to the distance.

In summary, data distance can affect cache performance for local queries, depending on the mobile client's movement and query patterns.

Valid scope area refers to the geometric area of the data instance's valid scope. It most accurately estimates access probabilities for different data instances of the same item when the user movement pattern is not known a priori. That is, the larger the data's valid scope area, the higher the probability that the client requests this data. This is because generally speaking, the client has a higher chance of being in large regions than in small ones. Thus, if we consider this factor only, it seems we should favor caching the data instances with larger valid scopes.

Manhattan Distance and Further Away Replace (FAR) are proposed data-distance-based cache replacement policies.[13,14] In these two policies, the data instance that is farthest from the client's current location is removed during replacement. In a recent study, we proposed two cache replacement policies that consider the valid scope area and data distance and combine them with access probability.[12]

In *data prefetching*, data is preloaded in the client cache for possible later use. For instance, if a user is going to pass a supermarket, the cache manager may prefetch shopping information. Given a limited cache size, the system requires intelligent prefetching to maximize the cache usage based on the user's speed and orientation.[15] Unfortunately, as with cache invalidation, existing work on cache replacement and prefetching only investigates simple queries. We expect that future studies will explore cache management issues for complex location-dependent queries.

The constraints of mobile computing environments, the spatial property of location-dependent data, and user mobility have opened new research problems for data management in LDISs. The issues presented in this article, while interesting, are by no means exclusive. We hope this article will stimulate discussion on LDIS data management issues. Many research opportunities in other aspects of LDISs, such as issues related to networking, and the provision of multimedia content, remain. P

ACKNOWLEDGMENTS

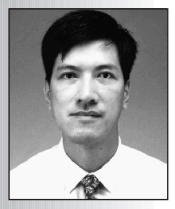
We thank the guest editors and the anonymous reviewers for their valuable comments and suggestions, which have improved the quality of this article. The Research Grant Council, Hong Kong SAR, China supported this research under grant number HKUST6079/01E.

REFERENCES

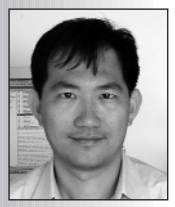
1. D. Barbara, "Mobile Computing and Databases: A Survey," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, no. 1, Jan./Feb. 1999, pp. 108–117.
2. A.Y. Seydim, M.H. Dunham, and V. Kumar, "Location Dependent Query Processing," *Proc. 2nd ACM Int'l Workshop Data Eng. for Wireless and Mobile Access* (MobiDE 01), ACM Press, New York, 2001, pp. 47–53.
3. J. Xu, B. Li, and D.L. Lee, "Placement Problems for Transparent Data Replication Proxy Services," to be published in *IEEE J. Selected Areas in Comm.*, 2002.
4. O. Wolfson, S. Jajodia, and Y. Huang, "An Adaptive Data Replication Algorithm," *ACM Trans. Database Systems*, vol. 22, no. 2, June 1997, pp. 255–314.
5. B. Zheng and D.L. Lee, "Processing Location-Dependent Queries in a Multicell Wireless Environment," *Proc. 2nd ACM Int'l Workshop Data Eng. for Wireless and Mobile Access* (MobiDE 01), ACM Press, New York, 2001, pp. 54–65.
6. V. Gaede and O.Gunther, "Multidimensional Access Methods," *ACM Computing Surveys*, vol. 30, no. 2, June 1998, pp. 170–231.
7. J. Xu, D.L. Lee, and B. Li, "On Bandwidth Allocation for Data Dissemination in Cellular Mobile Networks," to be published in *J. Wireless Networks*, ACM/Kluwer, 2002.
8. T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Data on Air: Organization and Access," *IEEE Trans. Knowledge and Data Eng.*, vol. 9, no. 3, May/June 1997, pp. 353-372.
9. T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Power Efficiency Filtering of Data on Air," *Proc. 4th Int'l Conf. Extending Database Technology* (EDBT 94), Springer, Heidelberg, Germany, 1994, pp. 245–258.
10. W.C. Lee and D.L. Lee, "Using Signature Techniques for Information Filtering in Wireless and Mobile Environments," *J. Distributed and Parallel Databases*, vol. 4, no. 3, July 1996, pp. 205–227.
11. J. Xu et al., "Cache Coherency in Location-Dependent Information Services for Mobile Environments," *Proc. 1st Int'l Conf. Mobile Data Access* (MDA 99), Springer, Heidelberg, Germany,1999, pp. 182–193.
12. B. Zheng, J. Xu, and D.L. Lee, "Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments," to be published in *IEEE Trans. Computers*, 2002.
13. S. Dar et al., "Semantic Data Caching and Replacement," *Proc. 22nd Int'l Conf. Very Large Data Bases* (VLDB 96), Morgan Kaufmann, San Fransisco, 1996, pp. 330–341.
14. Q. Ren and M.H. Dunham, "Using Semantic Caching to Manage Location Dependent Data in Mobile Computing," *Proc. 6th Ann. Int'l Conf. Mobile Computing and Networking* (MobiCom 2000), ACM Press, New York, 2000, pp. 210–221.
15. U. Kubach and K. Rothermel, "Exploiting Location Information for Infostation-Based Hoarding," *Proc. 7th Ann. Int'l Conf. Mobile Computing and Networking* (MobiCom 01), ACM Press, New York, 2001, pp. 15–27.

For more information on this or any other computing topic, please visit our Digital Library at http://computer.org/publications/dlib.

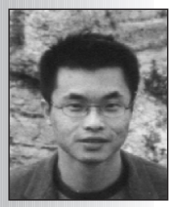
the AUTHORS



Dik Lun Lee is a professor in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests include document retrieval and management; discovery, management, and integration of information resources on the Internet; and mobile and pervasive computing. He received his MS and PhD in computer science from the University of Toronto. Contact him at the Dept. of Computer Science, Hong Kong Univ. of Science and Technology, Clear Water Bay, KLN Hong Kong; dlee@cs.ust.hk.



Wang-Chien Lee is an associate professor in the Computer Science and Engineering Department at Pennsylvania State University. His research interests include mobile and pervasive computing, data management, and Internet technologies. He received a PhD in computer and information science from the Ohio State University. He is a member of the IEEE Computer Society, IEEE Communications Society, and the ACM. Contact him at the Dept. of Computer Science and Eng., Pennsylvania State Univ., University Park, PA 16802; wlee@cse.psu.edu.



Jianliang Xu is an assistant professor in the Department of Computer Science at Hong Kong Baptist University. His research interests include mobile and pervasive computing, location-aware computing, Internet technologies, and wireless networks. He received a BEng in computer science and engineering from Zhejiang University, Hangzhou, China, and a PhD in computer science from Hong Kong University of Science and Technology. Contact him at the Dept. of Computer Science, Hong Kong Baptist Univ., Kowloon Tong, KLN Hong Kong; xujl@cs.ust.hk.

Baihua Zheng is a PhD candidate in the Department of Computer Science at the Hong Kong University of Science and Technology. Her research interests include mobile computing and spatial databases. She received a BEng in computer science and engineering from Zhejiang University, Hangzhou, China. Contact her at the Dept. of Computer Science, Hong Kong Univ. of Science and Technology, Clear Water Bay, KLN Hong Kong; email baihua@cs.ust.hk.