JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 30, 1187-1207 (2014)

DOI:10.1688/JISE.2014.30.4.15

# A Caching Strategy for Spatial Queries in Mobile Networks*

KWANGJIN PARK[1] AND YOUNG-SIK JEONG[2,+]
[1]*Department of Electrical Information Communication Engineering*
*Wonkwang University*
*Iksan, Chunbuk, 570-749 Korea*
[2]*Department of Multimedia Engineering*
*Dongguk University*
*Seoul, 100-715 Korea*

As a result of the recent developments in mobile terminals such as smartphones and laptop computers, as well as in wireless communication technologies such as GPS, location-based services are coming closer and closer to our daily life. However, the problems associated with the limited resources-such as limited wireless bandwidth, limited battery life, and small memory space-of mobile terminals remain unsolved. Research on cache utilization in mobile terminals, the development of efficient query processing algorithms, and improvement of the index structure in order to support efficient location-based service is geared towards alleviating these problems. In this paper, we propose location-based cache maintenance strategies for wireless broadcast environments in which a mobile client prefetches data that are expected to be used in the near future, and caches and maintains the data at a location close to the client's location. We also propose a hierarchical tree-based privacy approach for supporting anonymous location-based queries in wireless mobile data delivery systems. The results of experiments conducted using our proposed algorithms indicate that they help to reduce communication costs and support rapid spatial query processing.

***Keywords:*** cache replacement, moving objects, mobile computing, wireless data broadcasting, location-based services

## 1. INTRODUCTION

With the increasing availability of sophisticated mobile devices such as laptop computers and smartphones, users can gain access to a wide variety of services anytime and anywhere. Currently, smartphone handsets equipped with GPS are available; such phones enable easy and precise identification of the location of a user. Hence, there is an increased demand for mobile devices that offer location-based services (LBSs). In line with current technological developments, users can obtain a wide range of information about their locations [1-4, 20]. However, in the effort to efficiently support portable wireless lightweight devices, such as smartphones and laptop computers, loss of network connectivity and scarce resources (*e.g.*, battery life, processing power, and memory) are primary issues of concern. Data caching on mobile clients is considered an effective solution that improves system performance [16]. A client cache stores frequently accessed information so that queries can be answered without the need to connect to a server [10-14]. Even though data caching on mobile clients is considered an effective method for improving system performance [15], frequent disconnections and the inherent mobil-

Received August 29, 2012; revised January 7, 2013; accepted February 8, 2013.
Communicated by Jan-Jan Wu.
* This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (2013R1A1A1004593).
+ Corresponding author.

ity of clients may result in cache inconsistencies. In LBSs, the spatial properties of location-dependent data cause problems in data caching. For example, the cached result for a query (*e.g.*, the nearest restaurant) may become invalid when the client moves from one location to another [16]. When the valid scope of a data value is far from the client's current location, there is a very low probability of the data being suitable for reuse. Consequently, ejecting and replacing the farthest data may increase cache hits in the limited storage space. When cache hits increase, local data availability increases, and uplink and downlink costs and energy consumption decrease [16]. In this paper, we discuss mobile caching strategies that are effective when the service receiver (SR)-a client who receives services-is located near to the service provider (SP)-a client who provides services. These strategies can be extended to mobile computing environments, in which the range of wireless transmission initiated by the client is short and the resources available are limited, *i.e.*, low battery power, slow processors, and limited memory space. LBSs provide users with diverse, convenient service options, but have many risks in terms of customers' personal information and security [5]. Users are required to reveal their location in order to gain access to these services, *i.e.*, there is a tradeoff between privacy and convenience [6, 7]. Disclosing personal information such as location-related information can cause problems to the user in that all aspects of his/her private life may be monitored. For example, a certain company may abuse location information intended to be used as a marketing tool and obtain important information about its competitors. Another example is that it is possible to get information on the location of a specific person, check to see if there is anyone in the person's house, and subsequently commit a crime based on the information gleaned. Tracking an individual's activities without his/her consent is unethical, and hence, a high level of security is required for services that require users to disclose personal information [6, 9]. In this paper, we propose location-based cache maintenance strategies for mobile clients in wireless broadcast environments. The mobile client uses a wireless broadcast channel to prefetch data that are expected to be used in the near future and caches and maintains the data at a location close to the client's location. The proposed cache maintenance strategies can be categorized as follows:

- Nearest prefetch first: The SR stores data items that are located near to the current location.
- Hierarchical farthest away replace (HFAR): The SP removes data items that are located far away from the current location.

We also propose the use of a hierarchical tree structure to improve the efficiency of query processing. This proposal facilitates quick and precise query results by increasing the cache hit rate of the client. In addition, we define the privacy of a user and develop a search algorithm that meets user demand for privacy protection in wireless broadcast environments. A spatial index for the hierarchical tree structure is used to deliver the necessary information to the service provider, and the client selectively tunes the delivered information in order to obtain a desired final result without consuming a large amount of energy. To the best of our knowledge, this is the first work on supporting general location-dependent spatial queries for privacy protection in wireless broadcast environments. Our goal is to provide fast query processing with the minimum number of contacting nodes and selective tuning information in wireless broadcast environments.

The contributions made by this study can be summarized as follows:

- We propose cache maintenance strategies to support efficient spatial query processing in mobile computing environments. The proposed strategies predict data items that have a high possibility of being used in the near future on the basis of the location of the mobile client, and remove unnecessary data from the cache so that the efficiency of query processing is enhanced.
- We propose a hierarchical trees-structured index to facilitate efficient information search in wireless broadcast environments. The client can obtain accurate information within a short time by using a public minimal bounding rectangle (PMBR) index with a hierarchical structure in mobile computing environments.
- We define the privacy level of a user and propose the use of a search algorithm to meet user demand for privacy protection in wireless broadcast environments. Using this algorithm, in LBSs, the client sends location information in accordance with a level that corresponds to the user's demand for privacy protection. The client then obtains the exact results by query processing.
- We use synthetic and real data sets and conduct simulation experiments to validate the performance of our algorithms. Experimental results show that our proposed algorithms help to increase cache hits.

## 2. RELATED WORKS

In this section, we discuss work related to our proposed location-based query processing framework in three different areas: caching, location-based query processing, and privacy-aware query processing.

### 2.1 Client-Side Caching

When a user's location changes the results returned for a previous query may become invalid. Consequently, the client may have to tune the broadcast channel repeatedly [17]. This significantly increases not only the access time but also the energy consumed. Data caching has been proposed as a technique for reducing access latency and wireless bandwidth requirements. Prefetching can be performed without adding any load on shared resources such as wireless bandwidth [18]. In this technique, the client simply prefetches pages in anticipation of future accesses. In the context of mobile users, hoarding is a similar technique to prefetching for improving data availability. However, with mobile lightweight devices, such as laptop computers and smartphones, limited memory is a primary issue of concern [19, 20]. Due to the limited storage space available on mobile devices, an efficient cache replacement policy is vital to ensuring good cache performance. Moreover, the movement of mobile clients incurs new research problems for location-dependent query processing. In [37], the authors developed a mobility model to represent the moving behaviors of mobile users. They then investigated query processing and cache management strategies. In [32], the authors introduced a distance-based cache replacement method. In [11], the authors studied cache coherency issues in the context of LBS and proposed several cache invalidation schemes. For location-de-

pendent invalidation, they proposed bit vector with compression, grouped bit vector with compression, and implicit scope information methods. They also separated location-dependent data corruption from traditional cache corruption. In [21], the authors presented an indexing and semantic cache method for location-dependent queries based on Voronoi Diagrams (VDs). In this method, VDs are used to quickly preprocess the data in response to location dependent queries and a semantic cache is used to validate the answer. They also presented three cache replacement schemes based mainly on the size of the area and the distance between two centers of the cached data. In [38], the authors proposed cache management techniques for privacy preserving LBSs. The authors present three methods for cache replacement policies based on time, retrieval frequency, and mobile user density. Similar to LRU algorithm, the data, which has the largest time interval, will be discarded first for time based method. Next, retrieval frequency method (hereafter called VF) decides the weight of each data based on the number of times which it has been searched. The data with the lowest visit frequency will be replaced first. Finally, mobile User Density method discards low user density data. In [39], the authors proposed a cache management scheme using the concept of the geographically partial matching in the proxy server for location based services. They utilized the location-constraints to improve the cache efficiency and present a cache management that adopts the concept of the geographically partial matching for LBSs.

### 2.2 Location-Based Query Processing

Spatial databases have been studied over the last two decades, and several spatial database access methods have been proposed [3]. In [34], the authors presented a new system framework, called ROAD, for spatial object search on road networks. They also discussed the design, implementation, and evaluation of ROAD, and presented a holistic solution to several important research issues. In [35], the authors introduced a new data structure, called SD-tree, to preserve network connectivity and distance information for the duration of a query. They also proposed a continuous mobile network distance-based range query algorithm to support continuous range query processing with a large number of moving POIs (point of interests) in metro road networks. In [36], the authors presented an R-tree-based index[1] and query processing method for supporting geographic information systems. They introduced a multi-dimensional index structure to support higher efficiency query processing using a grid to process the data. The proposed method reduces spatial overlap and improves query efficiency by utilizing a cross-indexed file structure that is separate from the database system. In [26], the authors proposed a distributed index called DPTree that efficiently supports fulltext partial-match queries on P2P (peer-to-peer) networks. The idea is that each node manages a list of relevant documents for popular queries and organizes the document lists to be searchable within $O(\log N)$ time, where $N$ is the total number of participating nodes. They developed distributed pattern trees that selectively record query history and extend themselves by mining frequent patterns from the query history. In [27], the authors proposed the peer-to-peer R-tree (P2PR-tree), which is a spatial index specifically designed for P2P systems. They pointed out that existing R-tree-based structures are not adequate for P2P environments and proposed an R-tree1-based spatial index that is well suited to P2P environments. In [28], the authors illustrated how previous query results cached in the local storage of

[1] The R-tree is a classical spatial index structure. The basic idea is to approximate a spatial object with a minimal bounding rectangle (MBR) and to index the MBRs recursively.

neighboring clients can be leveraged to verify nearest neighbor queries at a local host. They identified a set of characteristics that enable the development of effective sharing methods and introduced a set of algorithms that aid in the decision process within this distributed environment; thus, they verified whether the data items received from neighboring clients provide a complete, partial, or irrelevant answer to the posed query. In [29], the authors investigated the problem of efficient execution of spatial queries in sensor networks. They proposed a peer-to-peer indexing structure called the peer-tree. They then presented a peer-to-peer query processing model over the peer-tree. The proposed model provides minimal energy consumption and response time with a distributed index structure. In [30], the authors studied multi-dimensional range queries in sensor networks, and introduced a new storage load balancing algorithm to achieve better balancing and reduce the latency and number of required operations.

### 2.3 Privacy-Aware Query Processing

Although mobile devices such as laptops, handheld PDAs, and smartphones coupled with location-based query processing promise safety and convenience, they threaten the privacy and security of users. With spatial query processing, a user trades his/her privacy for the benefits of the service [6]. In [22], the authors presented a systematic study on the problem associated with protecting location privacy under the network-constrained mobility model. They proposed a model for privacy-aware mobile services over road networks. The model provides a balance between the attack resilience of performance protection and the processing cost of anonymous queries while supporting a large number of mobile users with varying service requirements. In [9], the authors proposed a novel framework in which mobile users can entertain location-based services without the need to disclose their private location information. The proposed framework, called Casper, has two main components: a location anonymizer and a privacy-aware query processor. The location anonymizer functions as a trusted third party that blurs the exact location information of each user using a cloaked spatial area that matches the user privacy profile, while the privacy-aware query processor is embedded into traditional location-based database servers to tune their functionalities to be privacy-aware by dealing with cloaked spatial areas rather than exact point information. In [23], the authors presented a location cloaker and algorithms for processing *K*-nearest neighbor and range queries on spatial networks with privacy protection. The underlying idea is the concealment of the mobile user's exact location using a cloaked region. Spatial queries are executed based on both the cloaked region and the underlying networks and a candidate result set is returned to the requesting user, who filters out the exact answer. In [24], the authors presented a prototype system (called PROS) that can protect location privacy through peer-to-peer based cloaking on road networks. With PROS, users communicate with each other to find collaborative peers to achieve anonymity; the cloaked region is represented as a road segment set to exploit the features of road networks. In [25], the authors presented a framework for supporting anonymous location-based queries in mobile information delivery systems. They also provided a location privacy preference profile model called location P3P that allows mobile users to explicitly define their preferred location privacy requirements in terms of both location hiding and location QoS (quality of service) measures.

So far, we have introduced work related to our proposed location-based query processing framework in three different areas: caching, location-based query processing, and privacy-aware query processing. However, none of the above studies considered location-based data dissemination and indexing for spatial queries in periodic broadcast systems.

## 3. CACHE MAINTENANCE FOR FUTURE QUERIES

In this paper, we assume that the SR obtains the required information by using both a broadcast channel of sites and a point-to-point (on-demand) channel to contact sites[2]. In this section, we present adjacency data prefetching and cache replacement algorithms for use in LBSs. In adjacency data prefetching, the client prefetches the data objects that are adjacent to a particular location on a map for future use. We first review some problem characteristics that motivated our algorithm, then introduce the algorithm for HFAR replacement.

### 3.1 Problem Characteristics

Caching and prefetching have been successfully used to alleviate user-perceived latencies, and there has been extensive research in the area. In the prefetch mode, the client anticipates the information needs of users and tunes other clients for the required data [31]. Consequently, data caching and prefetching can be employed to improve access efficiency and alleviate the problem of limited wireless bandwidth in mobile environments. Data objects with very high access probabilities should be prefetched and cached in the client.

Let $t$ be a set of contacted sites (*e.g.*, information centers) and $T$ be a set of contacted sites and visited objects (*e.g.*, markets), where $t \subset T$.

**Definition 1** (Spatial query processing of the SR): Given a set of proved data objects SPD (Data object (*e.g.* hotel or gas station) that has verified its location from a client.) and a set of proved regions SPR (Area that is verified from a client, where SPD⊂SPR.), in $d$-dimensional geographic space $Rd$, retrieve object $O_i$ to query $Q$ (*i.e.*, range query $Q_r$ or nearest neighbor query $Q_{NN}$) according to the locations of objects from $t = \{s_1, s_2,.., s_t\}$, where $O_i \in$ SPD and SPD $\subset$ SPR.

The following running example illustrates this problem. Let us assume that client $C_1$ issues a query to obtain desired information, while client $C_2$ prefetches and caches desired information from close to the site.

**Example 1** (Effect of cache: nearest market searching): Suppose that a salesman wants to visit all markets that are close to the bus stops in a sales promotion effort (see Fig. 1). Assume that there are six markets near bus stops $a$ and $b$. In this running example, we do not consider wireless data broadcast. We assume that $S_1$ and $S_2$ store cached data items as follows: $S_1 = \{m_1, m_2, m_3\}$ and $S_4 = \{m_4, m_5, m_6\}$. Step (1): Salesman $C_1$ contacts all sites $S_1$, $S_2$, and $S_3$ at bus stop $a$ and issues a query such as "what are the names and addresses of the markets near bus stop $a$" to each site. In contrast, salesman $C_2$ prefetches data

[2] Let a site be a peer client providing spatial data or a static server, such as an information center.

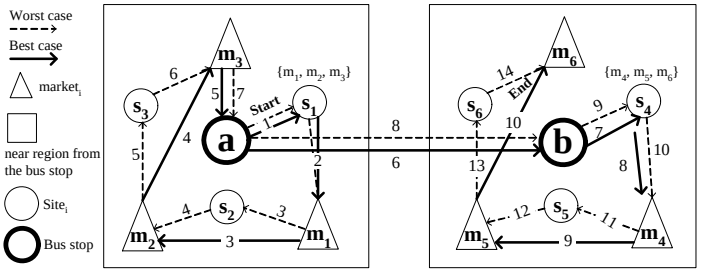



Fig. 1. An example of cache strategy for salesman.

items $m_1$, $m_2$, and $m_3$ from $S_1$ at bus stop $a$. $C_2$ caches the data items, then visits the markets sequentially without contacting sites $S_2$ and $S_3$. Step (2): Salesmen $C_1$ and $C_2$ move to bus stop $b$. Salesman $C_1$ contacts all sites $S_4$, $S_5$, and $S_6$ at bus stop $b$ and issues a query to each site.

In contrast, salesman $C_2$ prefetches data items $m_4$, $m_5$, and $m_6$ from $S_4$ at bus stop $b$. $C_2$ caches the data items, and then visits the markets sequentially without contacting $S_5$ and $S_6$. Consequently, $C_1$ obtains six markets after contacting six sites. We obtain the following values: $T = 6$, $t = 6$, and the number of places to visit = 14 with 6 request messages. On the other hand, $C_2$ obtains six markets by contacting sites $S_1$ and $S_4$. In this case, we obtain the following values: $T = 6$, $t = 2$, and the number of places to visit = 10 with no request messages.

**Example 2** (Effect of cache replacement: nearest market searching): Suppose that salesmen $C_1$ and $C_2$ want to visit markets that are close to the bus stops (see Fig. 2).

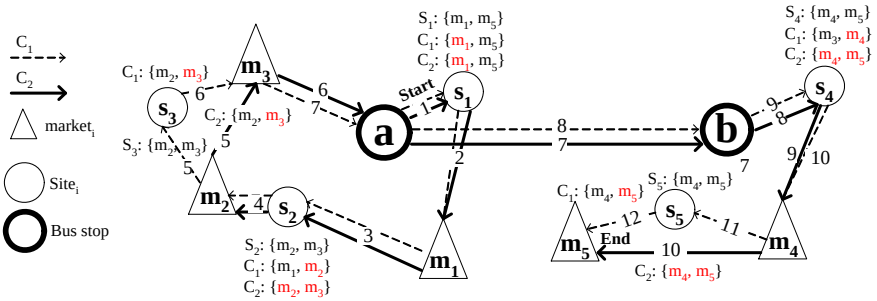



Fig. 2. An example of cache replacement strategy for salesman.

Assume that each site (*i.e.*, $S_1$,.., $S_5$) can only contain two entries pertaining to markets that are located near its location and a client prefetches data items from the nearest site in the initial step. A client issues a new query only if the cached result for a query is invalid. Assume that $C_1$ uses latest recently used (LRU) and $C_2$ uses HFAR cache maintenance strategies (which we will discuss in the next section). Step (1): $C_1$ and $C_2$ prefetch data items, such as "names and addresses of markets near the current location", that are cached in $S_1$. $C_1$ and $C_2$ then receive $m_1$ and $m_5$ from $S_1$ and cache them. The cached data items for both $C_1$ and $C_2$ are $\{m_1, m_5\}$. They then visit $m_1$. Step (2): $C_2$

prefetches data items $m_2$ and $m_3$ from $S_2$, removes $m_1$ and $m_5$ from the cache, and then stores $m_2$ and $m_3$ using HFAR; while $C_1$ still contains $m_1$ and $m_5$ from the cache using latest recently used (LRU). $C_1$ issues a query to $S_2$ because the cached result for a query in the position is invalid. The number of request messages from $C_1$ = 1. The cached data items for $C_1$ and $C_2$ are $\{m_1, m_2\}$ and $\{m_2, m_3\}$, respectively. $C_1$ and $C_2$ then visit $m_2$. Step (3): $C_1$ issues a query to $S_3$ and identifies $m_3$ from $S_3$, removes $m_1$ from the cache using LRU, and then stores $m_3$. $C_2$ does not need to contact $S_3$ since it already cached the desired data items. The number of request messages from $C_1$ is 2. The cached data items for both $C_1$ and $C_2$ are $\{m_2, m_3\}$. $C_1$ and $C_2$ visit $m_3$. Step (4): $C_1$ and $C_2$ move to bus stop *b*. $C_2$ prefetches data items $m_4$ and $m_5$ from $S_4$, removes $m_2$ and $m_3$ from the cache, and then stores $m_4$ and $m_5$. In contrast, $C_1$ issues a query to $S_4$ because the cached result for the corresponding query is invalid. The number of request messages from $C_1$ is 3. The cached data items for $C_1$ and $C_2$ are $\{m_3, m_4\}$ and $\{m_4, m_5\}$, respectively. $C_1$ and $C_2$ visit $m_4$. Step (5): $C_1$ issues a query to $S_5$ and identifies $m_5$ from $S_5$, removes $m_3$ from the cache using LRU, and then stores $m_5$. $C_2$ does not need to contact $S_5$ since it already cached the desired data items as in the previous steps. The number of request messages from $C_1$ is 4. The cached data items for both $C_1$ and $C_2$ are $\{m_4, m_5\}$. Finally, $C_1$ and $C_2$ both visit $m_5$. Consequently, for $C_1$, we obtain the following values: $t$ = 5 and the number of places to visit = 12 after 4 request messages. On the other hand, for $C_2$, we obtain the following values: $t$ = 3 and the number of places to visit = 10. Formally, the problem can be defined as follows. Given a verified set of data objects SPD, return the result for range query $Q_r$ and NN query $Q_{NN}$, with the minimal value of $t$, where $Q_r \subseteq$ SPR and $Q_{NN} \subseteq$ SPR. In summary, our objectives were to reduce response time by increasing the cache hits and to increase scalability by reducing the communication overhead between client and server.

### 3.2 Client-Side Cache Maintenance Strategy

In this section, we focus on a cache replacement policy to determine which item should be deleted from the cache when the cache does not have enough free space to accommodate a new item. We first present the public minimal boundary rectangle (PMBR) index tree that is maintained by a client. Then, we propose a novel cache replacement algorithm called hierarchical farthest away replacement (HFAR) that supports efficient selective tuning and improves cache hit rates. HFAR is designed to choose a victim data item that is furthest away from the client and remove it according to the hierarchical structure of an LBS. Assume that the client defines PMBR, a rectilinear shape that completely contains the bounded object(s) of each node (the objects can be gas stations, restaurants, *etc.*). The PMBR is constructed based on a minimal bounding rectangle (MBR), and the structure consists of three levels[3]: root-MBR (RMBR), sub-MBR (SMBR), and LEAF nodes. The base structure of the PMBR is similar to that of R-tree. However, unlike a conventional tree-based index, the entry in the internal node of each level does not contain a child-pointer pointing to a lower level node. Instead, the information broadcast by the server site is sequentially ordered based on the objects' location with a small amount of information[4]. The HFAR strategy first chooses the RMBR that is farthest from the user's current location *c*. The furthest SMBR and object is then sequentially evicted according to the distance from *c*. Suppose that the client who issues a query

[3] In this paper, we fixed the level of PMBR as three levels. However, the number of levels can be adjusted according to the number of objects.

[4] PMBR provides selective tuning (pruning and searching of entries) without pointers by sequential searching based on the location of each data object, thus resulting in a small index size.

tunes a broadcast channel of site *i*. Assume that the client prefetches the data objects that are located adjacent to each other in the map for future use if it has the free space to accommodate a new item. Since the information broadcast by the server site is sequentially ordered based on their location, clients can easily prefetch adjacent data objects while it processes NN or range search on air. Let *C* denote the set of cached items and *d* denote the value of threshold for the caching area, where the range of caching is the circle centered at *c* with radius *d*. For example, if the value of $d = 3$, the client prefetches items within a radius of three (miles) from its current location *c*, while it tunes the broadcast items from *n* sites. When the cache does not have enough free space to accommodate a new item, the victim item (the item furthest away-*i.e.*, RMBR, SMBR, or object-from the user's current location *c*) is hierarchically removed (from bottom to top) from the cache to maintain the value *d*. We assume that a node can store several RMBRs that may include SMBRs and objects, *e.g.*, $PMBR_1$, $PMBR_2$, ..., $PMBR_n$ if it has available free space for future queries. Before describing the algorithm in detail, we present an example.

Example (Cache replacement scenarios): Consider Fig. 3 (a) as an example. Suppose that the cache does not have enough free space to accommodate a new item. To select a cache tuple for replacement, $SMBR_4$ is first selected (has not yet been removed) since it is farther away from *c* than the other SMBRs. Objects $O_8$ and $O_7$ of $SMBR_4$ are removed for replacement. For the next victim data item, $SMBR_4$ is removed. The size of RMBR is diminished if $SMBR_4$ is removed, as shown in Fig. 3 (b). The order of victim data items is as follows: $O_8$, $O_7$, $SMBR_4$, $O_6$, $O_5$, $SMBR_3$, ..., *etc.*

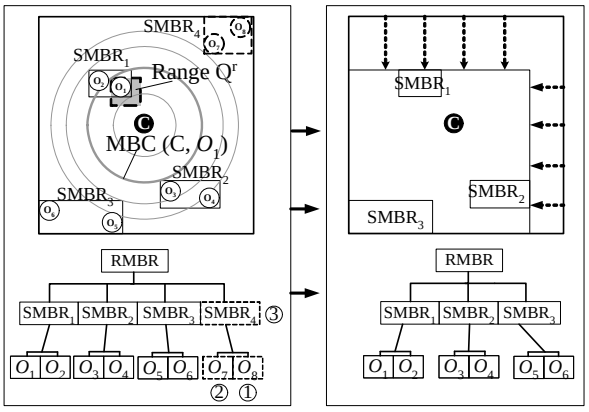



Fig. 3. An example for Hierarchical Farthest Away Cache Replacement; (a) Objects inside of $SMBR_4$, *i.e.*, $O_7$ and $O_8$, and $SMBR_4$ are selected for deletion from PMBR; (b) The reconstruction of PMBR after the deletion.

Note that the hierarchical order of the victim data items is important. The hierarchical structure of HFAR facilitates index searching by removing the data with the least access probability. Let us suppose the following two cases for range query $Q_r$ (shaded rectangle in Fig. 3 (a)). Case 1: $SMBR_4$ is removed for replacement. In this case, to check which $O_i$ is located inside of $Q_r$, the client has to tune and check all of the leaf data

items, such as $O_1$, $O_2$, ..., $O_8$. Case 2: HFAR replaces the victims, *e.g.*, $O_7$ and $O_8$, from the cache because they are furthest away from $Q_r$. To identify which $O_i$ is located inside of $Q_r$, the client only tunes the leaf data items $O_1$ and $O_2$ since the data items $SMBR_2$, $SMBR_3$, and $SMBR_4$ are out of MBC $(C, Q_r)$[5] while $O_1$ of $SMBR_1$ is located inside of $Q_r$. Fig. 4 shows the pseudo-code of HFAR.

**Algorithm:** Cache replacement
**Input:** cached data items
**Output:** victim data items
1: prefetch item according to the value of *d*
2: **if** (cache does not have enough free space) then
3: **while** (cache is full for prefetching) // to satisfy the value of *d*
4: check each $RMBR_i$ from cache
5: **if** $RMBR_i = \{\emptyset\}$ // All SMBRs and objects inside of $RMBR_i$ are already removed
6: then remove $RMBR_i$
7: **else**
8: check each SMBR from cache
9: **if** $SMBR_i = \{\emptyset\}$ // All object inside of $SMBR_i$ are already removed
10: then remove $SMBR_i$ from Cache
11: **else**
12: **for** each $O_i$ of the corresponding $SMBR_i$
13: find the farthest object $O_i$ from center and remove $O_i$

Fig. 4. Pseudo-code of HFAR.

Let us now look at the steps taken by the client to process the cache replacement. Let $O_v$ denote the victim object $O$ for replacement, where $O_v \in C$. Step (1): If cache memory has enough space to prefetch the desired item according to the value of *d*, then go to STEP (9); else, go to Step (2). Step (2): Check each RMBR from $C$ (an empty RMBR indicates that there is no candidate data object for the final result. Thus, the client deletes the empty RMBR first.) Step (3): If $RMBR_i = \{\emptyset\}$, then remove $RMBR_i$ and go to Step (1); else, go to Step (4). Step (4): Check each SMBR from $C$. Step (5): If $SMBR_i = \{\emptyset\}$, then remove $SMBR_i$ and go to Step (1); else, go to Step (6). Step (6): Scan $C$ and find the farthest object $O$ denoted as $O_v = \{O_v \in C \mid \forall O_i \in C: \text{dist}(O_v, c) > \text{dist}(O_i, c)\}$, where $O_v \neq O_i$. Step (7): Remove $O_v$ from $C$. Step (8): If the cache memory needs more space to prefetch the desired item according to the value of *d*, go to Step (1); else, go to Step (9). Step (9): Prefetch the desired item.

## 4. LOCATION ANONYMIZER WITH PMBR

In this section, we define the privacy level for a user and develop a search algorithm to meet user demand for privacy protection in wireless broadcast environments. We classify users' queries as listed below depending on the privacy level.

- Exact result with lower privacy: The user sends his/her exact location to the server. Based on the information sent, the user can obtain results with precise information on

[5] A minimum circle centered at $Q$ which contains $Q_r$.

the location. Therefore, the user's location is completely exposed. An example of this is "find the hotel that is closest to my location."

- Exact result with strict privacy: The user requests information on a location with a specific place as its starting point. As a result, the user can obtain precise information on a location without disclosing his/her location. An example of this is "find the gas station that is closest to city hall."
- Blurred result with strict privacy: The user obtains information on a location based on information he or she received from the SP without disclosing his or her location in a certain range. Thus, the user can protect his/her location information. However, the accuracy of the information decreases while the data items delivered from the SP increase in number. An example of this is "find any restaurant located in area A (rectangle or circle)." As a result, in a location-based service, the user sends the server location information at a level corresponding to his/her demand for privacy protection. This means that the more precise the location information a user sends to the server, the more exact the results the user will get. Conversely, the more location information the user conceals for privacy reasons, the more inaccurate the result he/she will receive. Imprecise results sent from the server require the client to conduct additional processing for the sake of information accuracy. In this section, we suggest a search algorithm that meets the user's demand for privacy protection in wireless broadcast environments by using a spatial index with a hierarchical structure.

### 4.1 The Role of the SP in Supporting Privacy-Aware Query Processing

In this section, we study the basic role of the SP and what the SP is required to send to the SR. The SP only has to deliver SMBR that includes or overlaps the query area to the SR. Based on the private region, the SP finds RMBRs, SMBRs, or corresponding objects before delivering them. If there is no RMBR that overlaps the private region, the SR expands the private region before sending it to the SR again.

Fig. 5 shows the private region that the SP received from the SR being checked and the hierarchical index tree before NN processing. The user location is the NN-query point. The SP cannot find the exact location of the query requester, as the SR delivers

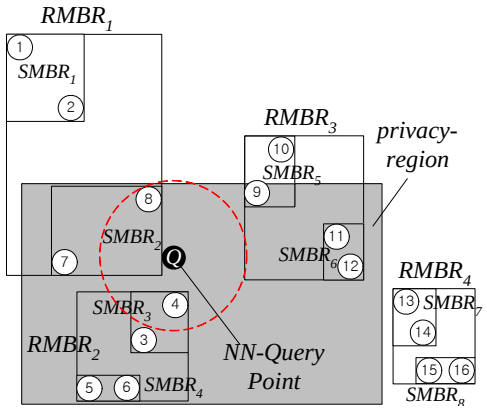



Fig. 5. SP query processing for NN search in private region.

only the defined private region to the SP. Below is the query processing procedure carried out by the SP. Step (1): The SP checks $RMBR_1$. Since $RMBR_1$ intersects the private region, the SP checks $SMBR_1$ and $SMBR_2$ of $RMBR_1$. Since $SMBR_2$ is located inside the private region, $SMBR_2$ is added to the result set = {$SMBR_2$}. Step (2): The SP checks $RMBR_2$. Since $RMBR_2$ is located inside the private region, $RMBR_2$ is added to the result set = {$SMBR_2$, $RMBR_2$}. Step (3): The SP checks $RMBR_3$. Since $RMBR_3$ intersects the private region, SP checks $SMBR_5$ and $SMBR_6$ of $RMBR_3$. Since $SMBR_5$ intersects the private region, the SP checks objects $O_9$ and $O_{10}$ of $SMBR_5$. Since object $O_9$ and $SMBR_6$ are located inside the private region, they are added to the result set = {$SMBR_2$, $RMBR_2$, $SMBR_6$, $O_9$}. Step (4): The SP checks $RMBR_4$. Since $RMBR_4$ is located outside the private region, the SP returns the final query result to the SR. The final result set= {$SMBR_2$, $RMBR_2$, $SMBR_6$, $O_9$}. We assume that a higher level node contains a lower level node in the HFAR index tree. For example, in the above example, $SMBR_2$ contains information on objects $O_5$ and $O_6$ in the final result set. We note that the SR may extend and resubmit the private region to the SP if it does not contain the desired result. That is, the SR resubmits its specified area, called 'private region', to the SP without exposing its exact location. The SR then obtains the final answer after pruning the processing result that is received from the SP. The SR determines the degree of location privacy by adjusting the private region. Fig. 6 shows the pseudo-code for the SP query processing.

**Algorithm:** SP Query Processing
**Input:** HFAR index
**Output:** result set that includes or intersects with the private region
Procedure:
1:**for** each $RMBR_i$
2: **if** ($RMBR_i$ is included in private region) then
3: add $RMBR_i$ to the result set
4: **else if** ($RMBR_i$ intersects with private region) then
5: **for** each $SMBR_i$ of $RMBR_i$
6: **if** ($SMBR_i$ is included in private region) then
7: add $SMBR_i$ to the result set
8: **else if** ($SMBR_i$ intersects with private region) then
9: **for** each object $O_i$ of $SMBR_i$
10: **if** $O_i$ is included in private region
11: add $O_i$ to the result set
12: return the result set to SR

Fig. 6. Pseudo-code for the SP query processing.

### 4.2 The Role of the SR in Supporting Privacy-Aware Query Processing

In this study, the SR creates an area with a certain range, called a private region-starting from the user's location (NN query point)-and sends the area to the SP. The SR freely determines the private region to use, including its size and the locations of the four vertices (see shaded box in Fig. 5). The SR is assumed to conceal its location in the private region and sends the region to the SP. Lack of a private region means that the user completely exposes his/her location. Even if the SR is the only person in the region from

which the request is coming, the SR can hide its exact location by adjusting the private region. In this section, we also define the process of tuning unclear location information that the SR receives from the SP. The SR processes the location information sent by the SP before obtaining the final result. In this case, selective tuning of the index delivered from the SP minimizes unnecessary energy usage. For the index search, a breadth-first order is assumed. The SR draws a minimal boundary circle from the object closest to PMBR or SMBR that the SR receives for the first time; and does not tune in for data outside of that region. RMBR or SMBR supports the SR's selective tuning because it has information on the arrival time for its lower-level node. For example, in the header, RMBR holds information on the arrival time of the first SMBR in the RMBR's lower level. The SR is assumed to tune the index in the following order (refer to the Fig. 5). Initial check list = {$SMBR_2$, $O_7$, $O_8$, $RMBR_2$, $SMBR_3$, $SMBR_4$, $O_3$, $O_4$, $O_5$, $O_6$, $SMBR_6$, $O_{11}$, $O_{12}$, $O_9$} (refer to query processing procedures taken by the SP) The SR's query processing procedure is as follows:

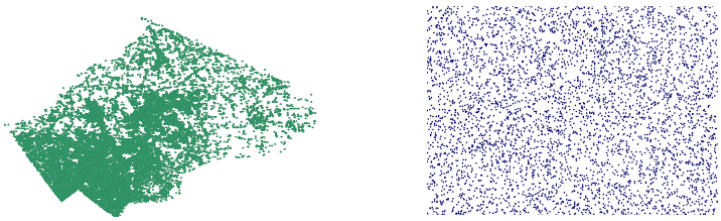
Step (1): The SR checks $O_7$ and $O_8$ of $SMBR_2$, and draws a minimal boundary circle (MBC) that contains $O_8$ because $O_8$ is currently located nearest to query $Q$. NNcandidate = $O8$. Remaining checklist = {$RMBR_2$, $SMBR_3$, $SMBR_4$, $O_3$, $O_4$, $O_5$, $O_6$, $SMBR_6$, $O_{11}$, $O_{12}$, $O_9$}. Step (2): The SR checks $RMBR_2$ since $RMBR_2$ intersects MBC. The SR also checks $SMBR_3$ since $SMBR_3$ intersects MBC. The SP identifies that $O_4$ is located inside of MBC. NNcandidate = $O_4$. Remaining checklist = {$SMBR_6$, $O_{11}$, $O_{12}$, $O_9$}. Step (3): The SP checks $SMBR_6$. $O_{11}$ and $O_{12}$ of $SMBR_6$ and $O_9$ are ignored because they are do not include or intersect MBC. NNcandidate = $O_4$. Remaining checklist = {Ø}. Step (4): The SP returns the final query result to the user. The final result is $O_4$. Fig. 7 shows the pseudo-code for SR query processing. In this study, we do not consider the range query processing algorithm because the range query is similar but much easier and simpler than the NN query.

**Algorithm:** SR Query Processing
**Input:** result set that includes or intersects with the private region
**Output:** result of nearest neighbor search
Procedure:
1:draw MBC from the first downloaded data object $O_i$
2: NN candidate = $O_i$
3:**for** each $RMBR_i$ //skip this step if the target check node is lower
4: **if** ($RMBR_i$ does not include or intersect with MBC) then
5: skip to download SMBRs of $RMBR_i$
6: **else**
7: **for** each $SMBR_i$ of $RMBR_i$ //skip this step if the target check node is lower
9: **if** ($SMBR_i$ does not include or intersect with MBC) then
10: skip to download the objects of $SMBR_i$
11: **else**
12: **for** each object $O_j$ of $SMBR_i$
13: **if** distance ($O_j$ , query point $Q$) < distance ($O_i$, query point $Q$) then
14: NN candidate = $O_j$
15: **else** NN candidate = $O_i$
16: return the final NN candidate to user

Fig. 7. Pseudo-code for the SR query processing.

## 5. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the HFAR framework with LRU and LFU. In processors, the doze mode has extremely low power consumption. We assume that the broadcast channel has a bandwidth of 2 Mbps. We used a uniform dataset (hereafter called $D_1$) with 10,000 points and a real dataset (hereafter called $D_2$) containing 39,231 data objects of MBRs for Montgomery County roads; the latter was extracted from the dataset available at [33] (see Figs. 8 (a) and (b)).



(a) REAL dataset ($D_1$). (b) UNIFORM dataset ($D_2$).

Fig. 8. Datasets for performance evaluation.

The SP generates the broadcast data stream with appropriate index information. Initially, the client stays in doze mode. When a mobile user sends a query, the client switches to active mode, starts to tune the broadcast channel, and accesses the data item according to the data access protocol. The client then switches to doze mode again if it has received the required information. The client prefetches pages in anticipation of future accesses. The probability distribution of the initial probe of the clients was assumed to be uniform within a broadcast, and all data items were considered to have the same size. The number of queries of the client while moving was limited to the range 10-50. In the experiment, the target data objects were uniformly distributed. The data objects were broadcast in a round-robin manner. In the performance evaluation, the number of cache hits was employed as the primary performance metric. Figs. 9 (a) and (b) show the results for the number of cache hits as the size of the cache increased in $D_1$ and $D_2$, respectively. As shown in the figures, the number of cache hits increased as the size of the cache increased.

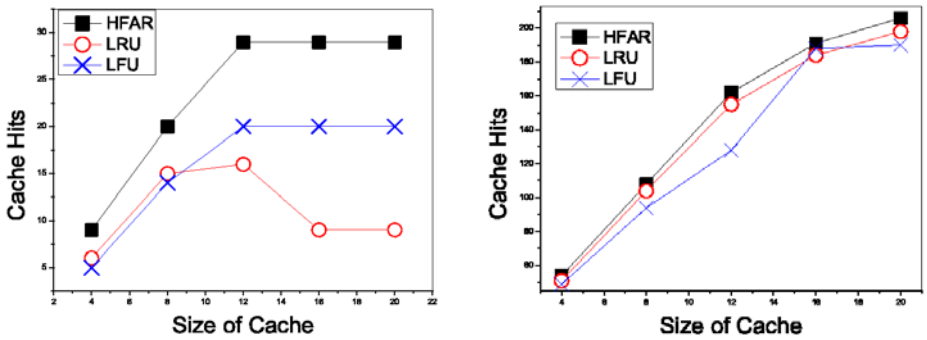



Fig. 9. The number of cache hits as the size of the cache increases in $D_1$ and $D_2$.

This is because the nodes were able to answer most of the client's queries if they had adequate cache capacity. It is clear to see that HFAR outperformed the other methods. This is because HFAR replaces the cached value based on the client's current location, while LRU and LFU replace the cached value based on the previous result. Clearly, if a client continuously moves its location, the client has a higher possibility of leaving certain cache hit regions. Consequently, the cached data are less likely to be reused for subsequent queries, which lead to worse performance.

Figs. 10 (a) and (b) show the results for the number of cache hits as the number of queries increased in $D_1$ and $D_2$, respectively. As expected, the performance of the replacement policies improved as the number of queries increased. In other words, the number of cache hits may increase as the number of queries increase within the same moving interval.

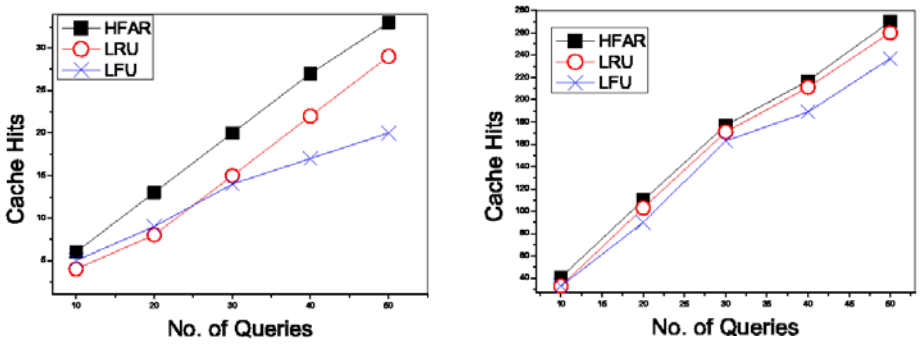



Fig. 10. The number of cache hits as the number of queries increases in $D_1$ and $D_2$.

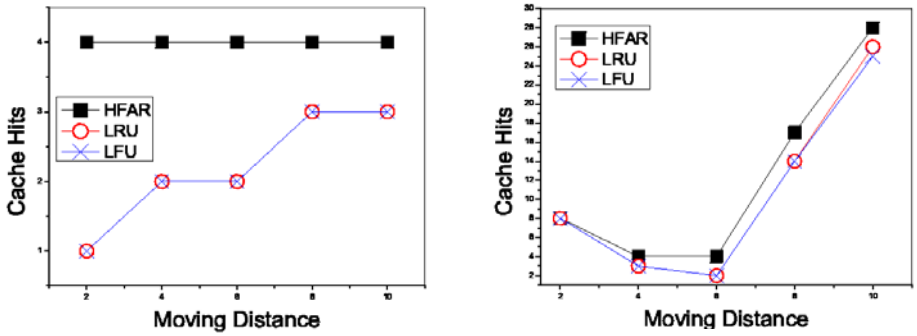



Fig. 11. The number of cache hits as the client's moving distance increases.

Figs. 11 (a) and (b) show the results for the number of cache hits as the client's moving distance increased in $D_1$ and $D_2$, respectively. As shown in the figures, HFAR outperformed LRU and LFU. This is mainly because the proposed algorithm preserves the possible result data for future queries in cache. In contrast, for LRU and LFU, the client maintains previously used data prior to the data pertaining to its current location.

Fig. 12 shows the results for the number of cache hits as the number of data points increased in $D_1$. It can be seen that HFAR outperformed the others, for the same reason outlined above. Next, we evaluate the number of cache hit with HFAR, VF (Visit frequency) and LRU. VF is the cache replacement policy to choose a victim data item that is the lowest visit frequency will be replaced first.

Figs. 13 (a) and (b) show the results for the number of cache hits as the update frequency is increased in $D_1$ and $D_2$, respectively. In the experiment, cached data will be updated when the amount (percent) of cached data is varied from 20% to 100%. As shown in the figures, HFAR outperforms VF and LRU. The main reason for this is that HFAR replaces the cached value based on the client's location, while VF and LRU replace the cached value based on the previous result.

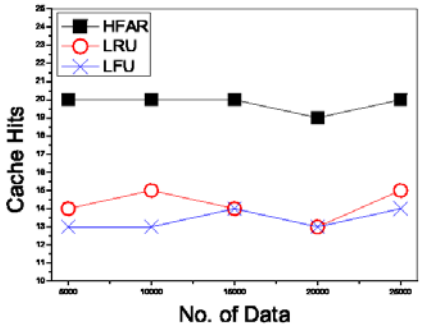



Fig. 12. The number of cache hits as the number of data increases in $D_1$.

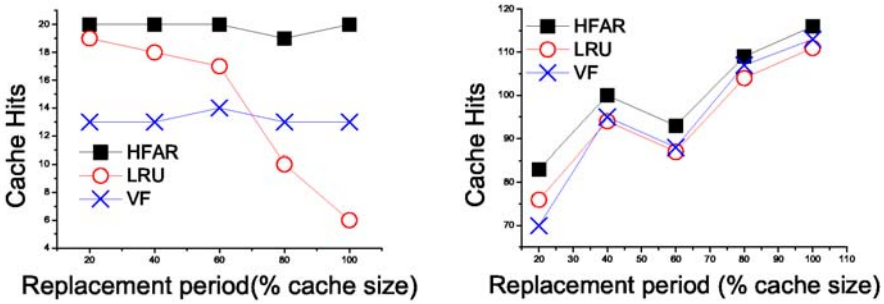



Fig. 13. The number of cache hits as the replacement period varies in $D_1$ and $D_2$.

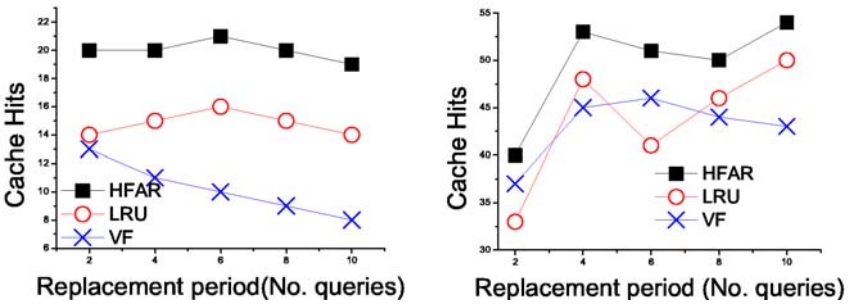



Fig. 14. The number of cache hits as the replacement period varies in $D_1$ and $D_2$.

Figs. 14 (a) and (b) show the results for the number of cache hits in $D_1$ and $D_2$, respectively. In the experiment, cached data will be updated according to the number of queries increases from 2 to 10. For the same reasons as above, HFAR exhibits the best result. Finally, we evaluated the access time and tuning time for evaluating the privacy-preserving technique of the range and NN queries as the size of the service area increased.

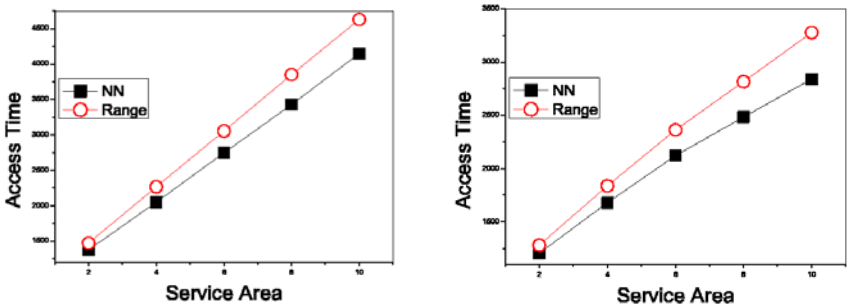



Fig. 15. The access time for NN and range queries as the size of the service area increases in $D_1$ and $D_2$.

Figs. 15 (a) and (b) show the results for the access time for NN and range queries as the size of the service area increased in $D_1$ and $D_2$, respectively. The percent of service area represents the percent of service coverage from the total service area for each node. For example, if we assume that the total service area is 100 km × 100 km, 10% of the service area indicates that the client performed spatial queries within a 10 km × 10 km area. Clearly, the client needs to access more data items as the percentage of service area increases.

Figs. 16 (a) and (b) show the tuning time results for NN and range queries as the size of the service area increased in $D_1$ and $D_2$, respectively. The results show that the larger service area (blurred result with strict privacy) incurred a large service overhead among the nodes. In summary, for spatial query processing, a user trades his/her privacy for better service.

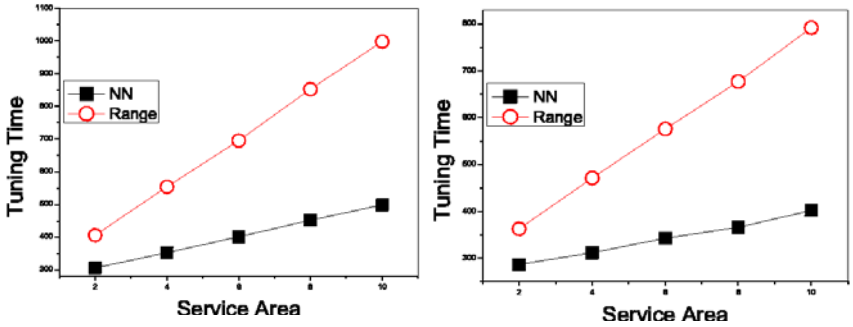



Fig. 16. The tuning time for NN and range queries as the size of the service area increases in $D_1$ and $D_2$.

## 6. CONCLUSION

In this paper, we presented a novel approach to reducing spatial query access latency and energy consumption in mobile computing environments. First, we proposed adjacency algorithms for data prefetching and cache replacement. In this method, the client maintains the data objects that are located adjacent to each other in a map in the cache for future use. As opposed to existing caching schemes, our caching algorithm determines cache replacement by choosing the victim data item that is farthest from the client and removing it according to the PMBR hierarchical structure. Ejecting the data that is farthest for replacement increases cache hits in the limited storage space. We next presented a hierarchical-tree-based privacy approach for supporting anonymous-location-based queries in wireless mobile data delivery systems. We designed two privacy-preserving query processing algorithms for the SP and the SR. The SR determines the privacy level and submits the private region to the SP, while the SP executes an algorithm based on the PMBR index and returns the result to the SR. The SR then selectively tunes the wireless channel and obtains the desired items; this helps in preserving the limited power resources. The proposed algorithms are simple but efficient in supporting linear transmission of spatial data and processing of location-aware queries. Since our technique involves the use of a cache maintenance algorithm, it helps to reduce communication overhead among nodes. Further, this technique can be used to obtain answers within a short time during query processing.

## REFERENCES

1. J. M. Kang, M. F. Mokbel, S. Shekhar, T. Xia, and D. Zhang, "Incremental and general evaluation of reverse nearest neighbors," *IEEE Transactions of Knowledge and Data Engineering*, Vol. 22, 2010, pp. 983-999.
2. J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Location-based spatial queries," in *Proceedings of Special Interest Group on Management of Data*, 2003, pp. 443-454.
3. K. Park, H. Choo, and P. Valduriez, "Scalable energy-efficient continuous nearest neighbor search in wireless broadcast systems," *Wireless Networks*, Vol. 16, 2010, pp. 1011-1031.
4. M. F. Mokbel, W. G. Aref, S. E. Hambrusch, and S. Prabhakar, "Towards scalable location-aware services: Requirements and research issues," in *Proceedings of ACM International Symposium on Advances in Geographic Information Systems*, 2003, pp. 110-117.
5. G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous location-based queries in distributed mobile systems," in *Proceedings of International World Wide Web Conference*, 2007, pp. 371-380.
6. M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: A privacy-aware location-based database server," in *Proceedings of International Conference on Data Engineering Conference*, 2007, pp. 1499-1500.
7. B. Gedik and L. Liu, "A customizable *k*-anonymity model for protecting location privacy," in *Proceedings of International Conference on Distributed Computing Sys-*

*tems*, 2005, pp. 620-629.
8. C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *Proceedings of ACM International Symposium on Advances in Geographic Information Systems*, 2006, pp. 171-178.
9. C.-Y. Chow, M. F. Mokbel, and W. G. Aref, "Casper*: Query processing for location services without compromising privacy," *ACM Transactions on Database Systems*, 2009, Vol. 34, p. 24.
10. K. Park and C.-S. Hwang, "Client-side caching for nearest neighbor queries," *Journal of Communications and Networks*, Vol. 7, 2005, pp. 417-428.
11. J. Xu, X. Tang, D. L. Lee, and Q. Hu, "Cache coherency in location-dependent information services for mobile environment," in *Proceedings of the 1st International Conference on Mobile Data Access*, 1999, pp. 182-193.
12. K. C. K. Lee, W.-C. Lee, J. Winter, B. Zheng, and J. Xu, "CS cache engine: data access accelerator for location-based service in mobile environments," in *Proceedings of Special Interest Group on Management of Data*, 2006, pp. 787-789.
13. D. Barbara and T. Imielinski, "Sleepers and workaholics: Caching strategies for mobile environments," in *Proceedings of Special Interest Group on Management of Data*, 1994, pp. 1-12.
14. G. Cao, "A scalable low-latency cache invalidation strategy for mobile environments," in *Proceedings of ACM/IEEE International Conference Mobile Computing and Networking*, 2000, pp. 200-209.
15. S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks: Data management for asymmetric communications environments," in *Proceedings of Special Interest Group on Management of Data*, 1995, pp. 199-210.
16. B. Zheng, J. Xu, and D. L. Lee, "Cache invalidation and replacement strategies for location-dependent data in mobile environments," *IEEE Transactions on Computers*, Vol. 51, 2002, pp. 1141-1153.
17. D. L. Lee, W.-C. Lee, J. Xu, and B. Zheng, "Data management in location-dependent information services: Challenges and issues," *IEEE Pervasive Computing*, Vol. 1, 2002, pp. 65-72.
18. V. Grassi, "Prefetching policies for energy saving and latency reduction in a wireless broadcast data delivery system," in *Proceedings of International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2004 pp. 77-84.
19. T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on air: Organization and access," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, 1997, pp. 353-372.
20. T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Energy efficient indexing on air," in *Proceedings of Special Interest Group on Management of Data*, 1994, pp. 25-36.
21. B. Zheng and D. L. Lee, "Semantic caching in location-dependent query processing," in *Proceedings of International Symposium on Spatial and Temporal Databases*, 2001, pp. 97-113.
22. T. Wang and L. Liu, "Privacy-aware mobile services over road networks," in *Proceedings of the 35th International Conference on Very Large Data Bases*, Vol. 2, 2009, pp. 1042-1053.
23. W.-S. Ku, Y. Chen, and R. Zimmermann, "Privacy protected spatial query pro-

cessing for advanced location based services," *Journal of Wireless Personal Communications*, Vol. 47, 2008, pp. 53-65.
24. J. Bao, H. Chen, and W.-S. Ku, "PROS: A peer-to-peer system for location privacy protection on road networks," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009, pp. 552-553.
25. B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting anonymous location queries in mobile environments with privacy grid," in *Proceedings of International World Wide Web Conference*, 2008, pp. 237-246.
26. D. Jing Zhao, D. Lun Lee, and Q. Luo, "DPTree, a distributed pattern tree index for partial-match queries in peer-to-peer networks," in *Proceedings of Extending Database Technology*, 2006, pp. 515-532.
27. A. Mondal, Y. Lifu, and M. Kitsuregawa, "P2PR-tree: An Rtree-based spatial index for peer-to-peer environments," in *Proceedings of Extending Database Technology Workshops*, 2004, pp. 516-525.
28. W.-S. Ku and R. Zimmermann, "Nearest neighbor queries with peer-to-peer data sharing in mobile environments," *Pervasive and Mobile Computing*, Vol. 4, 2008, pp. 775-788.
29. M. Demirbas and H. Ferhatosmanoglu, "Peer-to-peer spatial queries in sensor networks," in *Proceedings of International Conference on Peer-to-Peer Computing*, 2003, pp. 32-39.
30. G. Monti and G. Moro, "Multidimensional range query and load balancing in wireless ad hoc and sensor networks," in *Proceedings of International Conference on Peer-to-Peer Computing*, 2008, pp. 205-214.
31. W.-S. Ku, R. Zimmermann, and H. Wang, "Location-based spatial queries with data sharing in wireless broadcast environments," in *Proceedings of International Conference on Data Engineering*, 2007, pp. 1355-1359.
32. B. Xu, A. Ouksel, and O. Wolfson, "Opportunistic resource exchange in intervehicle ad hoc networks," in *Proceedings of the 5th IEEE International Conference on Mobile Data Management*, 2004, pp. 4-12.
33. http://www.rtreeportal.org.
34. K. C. K. Lee, W.-C. Lee, B. Zheng, and Y. Tian, "ROAD: A new spatial object search framework for road networks," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 24, 2012, pp. 547-560.
35. H. Wang and R. Zimmermann, "Processing of continuous location-based range queries on moving objects in road networks," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 23, 2011, pp. 1065-1078.
36. W. Zhang, X. Yang, W. Wu, and G. Xiang, "An optimized query index method based on R-tree," in *Proceedings of International Joint Conference on Computational Sciences and Optimization*, 2011, pp. 1007-1011.
37. Q. Ren and M. H. Dunham, "Using semantic caching to manage location dependent data in mobile computing," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, 2000, pp. 210-221.
38. Y. Chen, J. Bao, W.-S. Ku, and J.-L. Huang, "Cache management techniques for privacy protected location-based services," in *Proceedings of the 2nd International Workshop on Privacy-Aware Location-based Mobile Services*, 2008, pp. 1-6.

39. C.-M. Huang, M.-S. Lin, and W.-Y. Chang, "A novel cache management using geographically partial matching for location-based services," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 614-619.
40. K. Moon, N. Park, K. Chung, S. Sohn, and J. Ryou, "Trusted certificate validation scheme for open LBS application based on XML web services," *Journal of Information Processing Systems*, Vol. 1, 2005, pp. 86-95.

**Kwangjin Park** received the Ph.D. degree in Computer Science from Korea University in 2006. In 2008, he joined the Schools of Electrical Electronics and Information Engineering, Wonkwang University, where he is an Associate Professor. His research interests include spatiotemporal databases, mobile databases, and data dissemination.

**Young-Sik Jeong** is a Professor in the Department of Multimedia Engineering at Dongguk University in Korea. His research interests include grid and cloud computing, mobile and ubiquitous sensor network (USN) computing, and USN middleware. He received his B.S. degree in Mathematics and his M.S. and Ph.D. degrees in Computer Science and Engineering from Korea University in Seoul, Korea in 1987, 1989, and 1993, respectively. Since 1993, he has been serving as an IEC/TC 100 Korean Technical Committee member, as the IEC/TC 108 Chairman of Korean Technical Committee, and as an ISO/IEC JTC1 SC25 Korean Technical Committee member. He is also is a member of the IEEE.