Location-Based Services for Dynamic Range Queries

Kwangjin Park, Moonbae Song, and Chong-Sun Hwang

***Abstract:* To conserve the usage of energy, indexing techniques have been developed in a wireless mobile environment. However, the use of interleaved index segments in a broadcast cycle increases the average *access latency* for the clients. In this paper, we present the broadcast-based location dependent data delivery scheme (BBS) for dynamic range queries. In the BBS, broadcasted data objects are sorted sequentially based on their locations, and the server broadcasts the location dependent data along with an index segment. Then, we present a data prefetching and caching scheme, designed to reduce the query response time. The performance of this scheme is investigated in relation to various environmental variables, such as the distributions of the data objects, the average speed of the clients, and the size of the service area.**

***Index Terms:* Air indexing, caching, data broadcasting, location-based services, range queries.**

I. INTRODUCTION

Recently, the efficient storage and retrieval of moving objects in database management systems has attracted considerable attention. In location-aware mobile services (LAMSs), the server is characterized by the large number of mobile clients and stationary objects that they have to manage. In this environment, both mobile and stationary clients have the ability to issue spatial queries. The broadcasting of spatial data is an effective way of disseminating data in a wireless mobile environment, since this method can be scaled up without any penalty being incurred, when the number of users grows. However, performance of the query processing obtained with a broadcasting method is highly dependent on the order in which the data is broadcasted. Therefore, the question of how to organize the sequence of the broadcast data is a very important issue [1].

In the broadcast-based model, the broadcasting of data together with an index structure is an effective way of disseminating data in a wireless mobile environment [2]. Using an index can help the client to reduce the amount of time spent listening to the broadcast channel. However, the average time which elapses between the request for the data and its receipt may be increased as a result of these additional messages. Air indexing techniques can be evaluated in terms of the following factors.

- *access latency*: The average time elapsed from the moment a client issues a query to the moment when the required data item is received by the client (see Fig. 1).
- *tuning time*: The amount of time spent by a client listening to the channel (see Fig. 1).

The *access latency* consists of two separate components, namely,

- *probe wait*: The average duration for getting to the next index segment. If we assume that the distance between two consecutive index segment is L, then the *probe wait* is $L/2$.
- *Bcast wait*: The average duration from the moment the index segment is encountered to the moment when the required data item is downloaded.

The *access latency* is the sum of the *probe wait* and *Bcast wait*, and these two factors work against each other [3], [4].

In general, the fastest access time in a broadcast cycle is obtained when there is no index, but this increases the *tuning time*. On the other hand, increasing the number of index segments in a single broadcast cycle reduces the *tuning time*, but increases the *access latency* [3], [4]. Therefore, the number of indices in the broadcast cycle has to be optimized by taking into consideration the *access latency*. Consequently, in order to achieve efficient indexing on air, it is necessary to simultaneously minimize both the *probe wait* time and *access latency* by adjusting the number of indices in the broadcast cycle.

In this paper, we aim to provide research directions towards reducing both the *tuning time* and *access latency* for LAMSs. Let's consider an example in which a user driving in his or her car and sends a query, such as, "As I am moving in a certain direction, show me all gas stations within 10 km of my location." According to the user's location and the size of the query range, the result will be dynamically changed. To handle such a query, the positions of the objects and the mobile client must be found. For location-aware range queries, we first introduce the broadcast-based location dependent data delivery scheme (BBS). In this scheme, the server periodically broadcasts reports, which contain the IDs of the data objects (e.g., building names) and their location coordinates, to the clients. These broadcasted data objects are sorted sequentially based on their location before being broadcasted. Then, we introduce a prefetching scheme for use with dynamic range queries in LAMSs. The main contributions of our work can be summarized as follows.

- The client can perform range query processing, even if the desired data objects arrive before the associated index segment is received. This technique significantly reduces the latency in broadcast-based location-aware query processing.
- The client simply adjusts the value of r when performing dynamic range query processing.
- The client can also perform range query processing without an index segment. In this case, the best access time is obtained, since no index is broadcast along with the file [3], [4].

The remainder of the paper is organized as follows. Section II provides background information on the index model and cache maintenance scheme. Section III discusses the problem description and Section IV describes the proposed BBS scheme and prefetching method. A performance evaluation is presented in Section V. Finally, Section VI concludes this paper.

Manuscript received December 7, 2004; approved for publication by Thomas LaPorta, Division II Editor, July 5, 2005.

The authors are with the Department of Computer Science and Engineering, Korea University, 5-1, Anam-dong, Seongbuk-Ku, Seoul, 136-701, Korea, email: {kjpark, mbsong, hwang}@disys.korea.ac.kr.

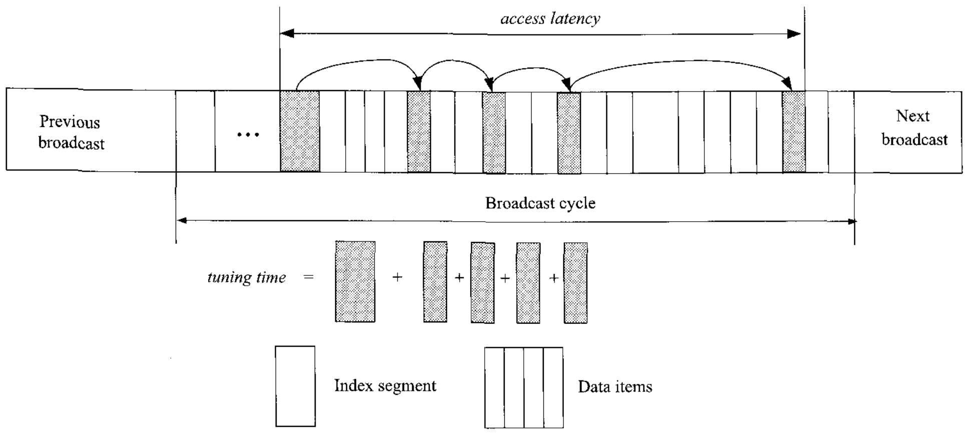


Fig. 1. *access latency* and *tuning time* in wireless broadcast.

II. BACKGROUND

With the advent of high speed wireless networks and portable devices, data requests based on the location of mobile clients have increased in number. However, there are several challenges to be met in the development of location aware query processing [5], such as the constraints associated with the mobile environment and the difficulty of taking the user's movement into account. Hence, various techniques have been proposed to overcome these difficulties.

A. Spatial Data Services

In [2], authors present a new index structure, called D-tree, which indexes spatial regions based on the divisions that form the boundaries of the regions. Different from the existing approaches, the D-tree neither decomposes nor approximates data regions; rather, it indexes them directly based on the divisions between the regions. The basic idea of the D-tree is to index data regions based on their boundaries. In [6], authors present new index technique and search algorithms to fit the sequential access property of wireless data broadcast. The proposed scheme replicates multiple search trees into linear structure and distributes index structure over the whole broadcast cycle. The main idea is to allow a client to start query processing as soon as possible in order to reduce the *access latency*. In [7], authors present algorithms to perform KNN search on conventional sequential-access R-tree. Authors point out that commonly employed depth first tree results in reading many nodes and thus, causing higher energy consumption. They present algorithms to adapt and KNN search on R-tree family, instead of introducing another indexing structure. Furthermore, they investigate the use of histograms as a technique to improve tune-in time and memory requirement of KNN search. In [8] and [9], a linear index structure is proposed based on the Hilbert curve, in order to enable the linear broadcasting of objects in a multi-dimensional space. They address the issues involved with organizing location dependent data and answering spatial queries on air. However, the Hilbert curve needs to allocate a sufficient number of bits to represent the index values, in order to guarantee that each of the points in the original space has a distinct value. If k is the number of bits used for a coordinate in the ith dimension of the targeted m-dimensional space and n is the number of bits assigned to represent the coordinates, then a total of $\sum_{i=1}^{m} k$ bits need to be allocated to represent the coordinates and the expected time for the conversion is $O(n^2)$. Besides, with this scheme, in order to identify the sequence of the broadcast data items, the clients have to wait until the index segment is arrives, even if the desired data is just in front of them. In [10], authors discuss the specific characteristics of broadcast environments and conclude that existing index structures are unsuitable for wireless broadcast environment, since the most of the previous studies do not consider the time-series characteristics of the air index. Authors address the issues involved with organizing location dependent data and answering KNN queries on air. Then, a new index structure, called sorted list is proposed to enable a linear transmission of location dependent data and processing of KNN queries. However, they only provide approximate search techniques. In [11], authors present techniques for scheduling a spatial index tree for broadcast in a single and double channel environment. The algorithms executed by the clients aim to minimize latency and *tuning time*.

In the mobile computing environment, caching data at the client's side is a useful technique for improving the performance. However, the frequently disconnection and mobility of the clients may cause cache inconsistency problems. In [12], authors propose location dependent cache invalidation schemes for mobile environments. In this scheme, they use bits to indicate whether the data item in the specific area has been changed. For instance, if there are eight service areas and the values of the bit vector are `00010011`, this means that the data item is valid in 4-th, 7-th, and 8-th only, and they organize each service area as a group in order to reduce the overhead for scope information. In [13], authors proposed a polygonal endpoint (PE) and approximate circle (AC) schemes. The PE scheme records all the endpoints of the polygon representing the valid scope, while the AC scheme uses an inscribed circle from the polygon to represent the valid scope of the data.

B. Broadcast Model

Disseminating data through a broadcast channel allows simultaneous access by an arbitrary number of mobile users and thus allows efficient usage of scarce bandwidth. In [14], authors in-

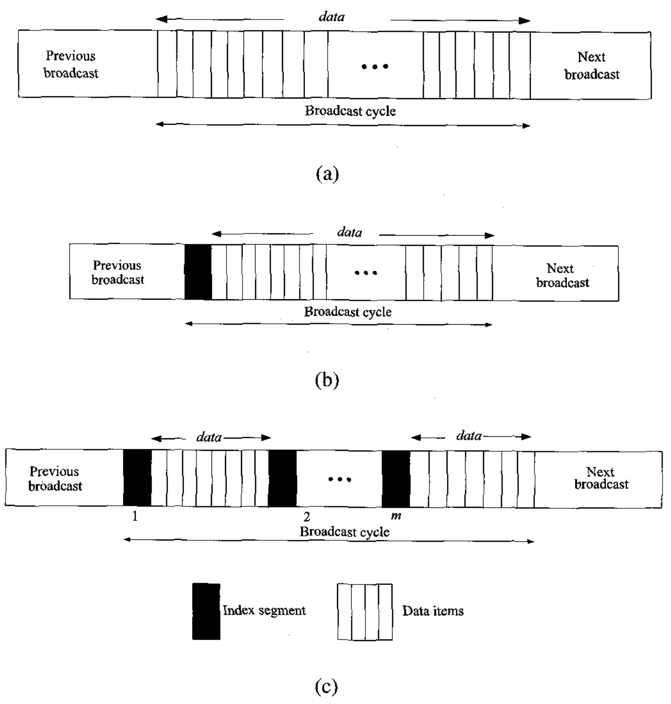


Fig. 2. One dimensional optimal methods: (a) opt_access, (b) opt_tune, (c) $(1, m)$ index.

troduce a technique for delivering data objects to the clients in asymmetric environments. In this scheme, groups of pages, such as hot and cold groups, with different broadcast frequencies are multiplexed on the same channel. Then, items stored on the faster disks are broadcast more often than items on the slower disks. The broadcast disk technique has two main components. First, multiple broadcast programs with different latencies are superimposed on a single broadcast channel, in order to provide improved performance for non-uniform data access patterns and increased availability for critical data. Second, the technique integrates the use of client storage resources for caching and prefetching data that is delivered over the broadcast.

C. Index on Air

Data broadcasting in a wireless network constitutes an attractive approach in the mobile data environment. However, the wireless broadcast environment is affected by the narrow network bandwidth and the battery power restrictions of the mobile clients. The broadcasting of spatial data together with an index structure is an effective way of disseminating data in a wireless mobile environment. This method allows mobile clients requesting data to tune into a continuous broadcast channel only when spatial data of interest and relevance is available on the channel, thus minimizing their power consumption. Broadcasting methods that properly interleave index information and data on the broadcast channel can significantly improve not only energy efficiency, but also *access latency*. Let *data* be the number of the data objects and C be the download time for required records. There are two parameters that need to be optimized in the one dimensional space of the *access latency* and the *tuning time* [3], [4].

- Optimal_Latency: The best latency is obtained when no index is broadcast along with the data. In this case, the size of the entire *Bcast* is minimized, and the average latency time is $\frac{data}{2} + C$, however the worst value of the average *tuning time* is obtained, since it is equal to $\frac{data}{2} + C$, as shown in Fig. 2(a).
- Optimal_Tune: This parameter allows the best *tuning time* to be obtained, while increasing the *access latency*. The server broadcasts the index at the beginning of each *Bcast*, as shown in Fig. 2(b). In this case, *probe wait* is equal to $\frac{data+index}{2}$ and the *Bcast wait* is equal to $\frac{data+index}{2} + C$.
- $(1, m)$ index: In this method, the index is broadcast m times during a single broadcast cycle. The broadcast index is broadcasted every fraction $(\frac{1}{m})$ of the broadcast cycle. In order to reduce the *tuning time*, each index segment and each data contains a pointer pointing to the root of the next index. In case of Optimal_Tune and $(1, m)$ indexing, selective tuning is accomplished by multiplexing an index with the data items in the broadcast. The clients are only required to operate in active mode when probing for the address of the index and downloading the required data items, while spending the waiting time in power save mode (see Fig. 2(c)).

Since the latency is the sum of the *probe wait* and the *Bcast wait*, the average latency time is equal to $(\frac{data+index}{2})2 + C = data + index + C$. There are several indexing techniques such as the distributed indexing approach [4], the signature approach [15], and the hybrid approach [16]. In this paper, we employ the $(1, m)$ interleaving technique, in order to interleave the index and the data on the broadcast channel.

III. PROBLEM DESCRIPTION

An index gives the ability of selective tuning. By accessing the index segment, the client is able to predict the arrival times of the desired data items. Then, the client tunes into the broadcast channel when the desired data items arrive and downloads the data items. With this purpose, the client search the sparse index tree to obtain the approximate location information about the desired data items. However, existing indexing structures are unsuitable for LAMSs in wireless data broadcast. The major problems which we have to consider are three-fold. First, the broadcast cycle is lengthened due to the additional messages. The longer broadcast cycle increases, the average *access latency*. Second, the effort of searching for retrieving the arrival time of the desired data items increases the *tuning time* and *access latency*. The existing search algorithms are designed for traditional spatial database and do not consider the time-series characteristics of the air index. For example, R-tree and its variant such as R+-tree or R*-tree have been widely used to support various spatial queries. Thus, mostly existing algorithms proposed for location-based query processing are based on R-tree. However, backtracking may incur significant *access latency*. R-tree-based methods are better supported by random access storages, such as memory and disk, but not the wireless channels. Third, although D-tree [2], Hilbert-curve index structure [8], [9], and distributed spatial index (DSI) [6] alleviate backtracking problem, this method has the worst latency because clients have to wait until the beginning of the next broadcast cycle in order to obtain index segment, even if the desired data is just in front of

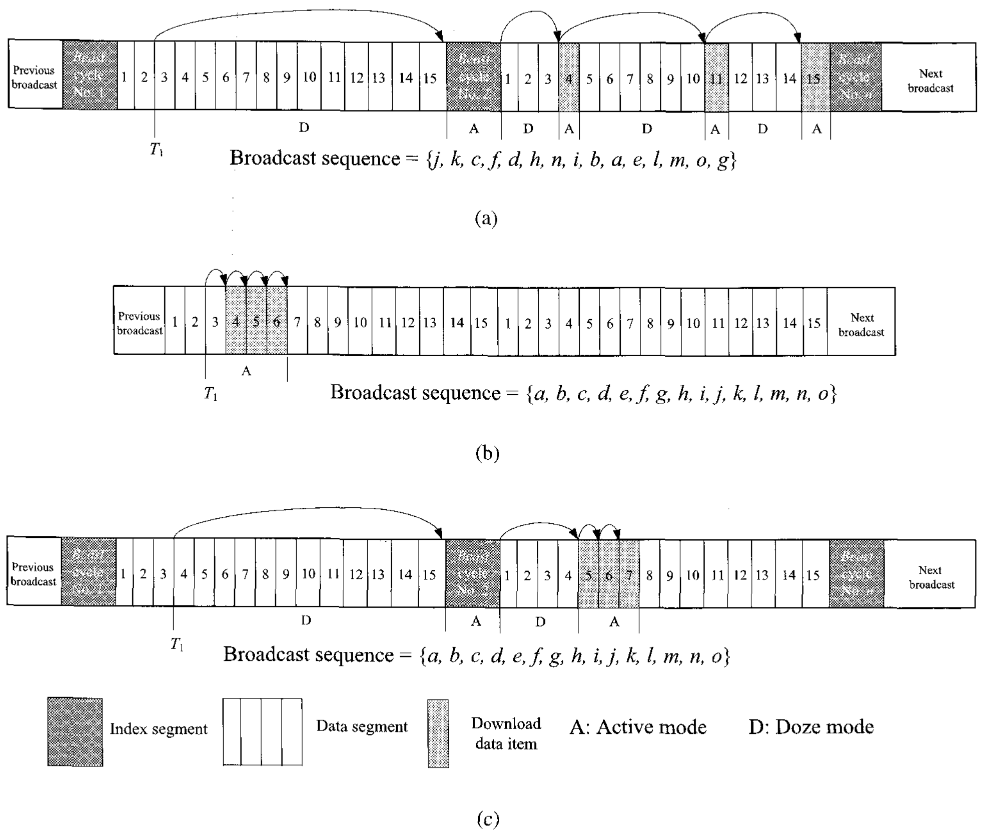


Fig. 3. One dimensional data broadcast: (a) $(1, m)$ index, (b) BBS with opt_access, (c) BBS with opt_tune.

them. Thus, search algorithms need to be revised to fit the linear streaming property of wireless data broadcast. The client unable to process spatial queries without an index segment. If the client trying to process spatial queries without index segment, it has to tune the whole broadcast cycle. Therefore, this methods have the worst possible latency, because the clients have to wait until the beginning of the next broadcast cycle, even if the desired data is just in front of them. Thus, we conclude that the existing indexing methods are unsuitable for LAMSs.

Let's consider the three examples of Fig. 3, in which the client is desired to obtain the data item e, f, g.

Example 1: We assume that an index is broadcast along with data items and the client begins to tune the broadcast channel at T_1. In this case, the client has to wait until the next broadcast cycle (e.g., *Bcast* cycle 2 in Fig. 3(a)), even the desired data items e, f, g (i.e., 11-th, 4-th, and 15-th data items, respectively) are broadcasted in the current broadcast cycle. Since the *access latency* is the sum of the *probe wait* and the *Bcast wait*, we obtain following result for Example 1: *probe wait* $= 13$ and *Bcast wait* $= 15$, then the total *access latency* $= 28$.

Example 2: We assume that no index is broadcast along with data items and the client begins to tune the broadcast channel at T_1 as shown in Fig. 3(b). In this case, the client simply tune into the broadcast channel and filter all data till the required data items are downloaded. Since the client dose not need to wait and tune into the broadcast channel to receive an index, it dose not need to add the *probe wait* to calculate the total *access latency*. *Bcast wait* $= 4$, then total access *latency* $= 4$.

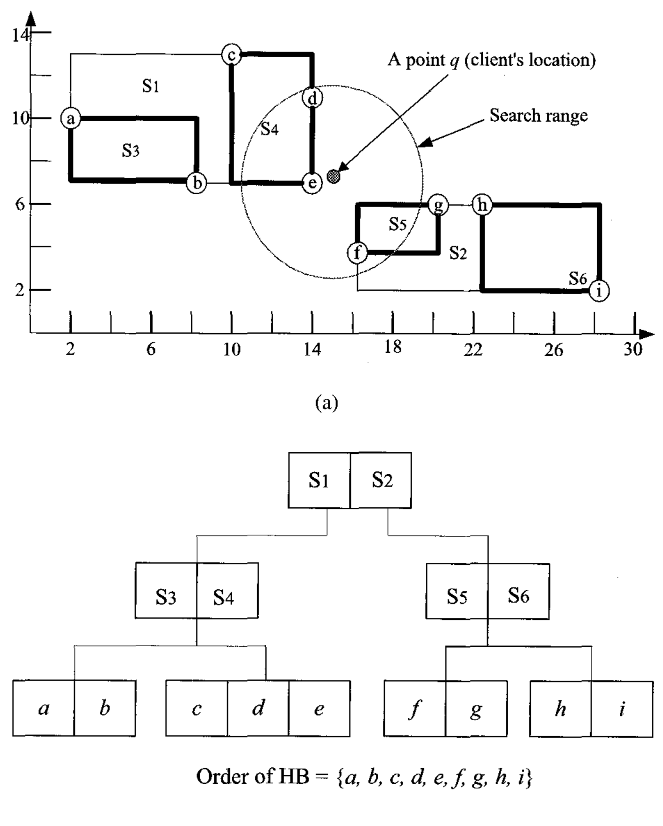
Example 3: This is the same as the Example 1, except the broadcast data items are sequentially ordered based on their location as shown in Fig. 3(c). The client has to wait for an index, in order to process the range queries in the same way of the Example 1. Since the *access latency* is the sum of the *probe wait* and the *Bcast wait*, we obtain following result for Example 3: *probe wait* $= 13$ and *Bcast wait* $= 7$, then total *access latency* $= 20$.

IV. PROPOSED ALGORITHMS

In this section, we describe two schemes for use in conjunction with LAMSs. We first introduce the broadcast-based location dependent data delivery scheme (BBS). In this scheme, the server broadcasts reports which contain the IDs of the data objects (e.g., building names) and the values of the location coordinates. The data objects broadcast by the server are sorted based on the locations of the data objects. Then, we present a sequential prefetching scheme and object boundary circle (OBC), in order to reduce the client's *tuning time*. In this paper, we assume a geometric location model, i.e., a location is specified as a two-dimensional coordinate and the broadcasted data objects are static, such as restaurants, hospitals, and hotels. Mobile clients can identify their locations using systems such as the global positioning system (GPS).

A. Broadcast Based LDIS (BBS) Scheme

Through the insertion of an index in the broadcast cycle, the mobile client is able to predict the arrival time of the desired data object, as well as being able to perform selective tuning.



A point q (client's location)
Search range
S1
S2
S3
S4
S5
S6
a
b
c
d
e
f
g
h
i
(a)
(b)
Order of HB = {a, b, c, d, e, f, g, h, i}

Fig. 4. Example of horizontal broadcasting: (a) Minimun boundary rectangle, (b) R-tree.

The drawback of this solution is that the client has to wait and listen for an index segment, in order to identify the desired data object. As stated earlier, when without an index, the average time of latency is $\frac{data}{2}$. However, to handle the nearest neighbor (NN)-search and the range query processing, average time of latency is equal to $data$, since the client has to check the whole data objects to identify the NN data. In this section, we propose a technique designed to reduce the *access latency*.

In the BBS method, the sever periodically broadcasts the IDs and the coordinates of the data objects, along with an index segment, to the clients, and these broadcasted data objects are sorted sequentially according to the location of the data objects before being sent to the clients. In the BBS method, since the data objects broadcasted by the server are sequentially ordered based on their location, it is not necessary for the client to wait for an index segment, if it has already identified the desired data object before the associated index segment has arrived. In this method, the structure of the broadcast affects the distribution of the data object. For example, as shown in Fig. 4, if the data objects are horizontally distributed, the server broadcasts the data objects sequentially, from the leftmost data object to the rightmost data object. A simple sequential broadcast can be generated by linearizing the two dimensional coordinates in two different ways, i.e., horizontal broadcasting (HB) or vertical broadcasting (VB). In HB, the server broadcasts the LDD (location dependent data) in horizontal order, that is, from the leftmost coordinate to the rightmost coordinate, as shown in Fig. 4. On the other hand, in VB, the server broadcasts the LDD in vertical order, that is, from the bottom coordinate to the top coordinate. In order to decide whether to broadcast the data using HB or VB, the server uses the following algorithm.

Notations:

- *leftmost_P*: A point that is located at the leftmost extremity in the map (e.g., object 'a' in Fig. 4)
- *leftmost_P'*: The x-axis coordinate of a point that is located at the leftmost extremity in the map with the exception of *leftmost_P*, where the value of the coordinates of *leftmost_P'* $\geq$ *leftmost_P* (e.g., object 'b' in Fig. 4)
- *x-dist*: Distance between *leftmost_P* and *leftmost_P'* based on x-axis
- *y-dist*: Distance between *leftmost_P* and *leftmost_P'* based on y-axis
- *x-dist_counter*: Initial value is 0
- *y-dist_counter*: Initial value is 0
- *MAX*: Number of data objects
- *NOC*: Number of compares (initial value is 0)

Algorithm 1. The server decision algorithm for VB or HB data broadcasting

Input: Data objects' IDs and locations;
Output: Selection result for HB or VB;
Procedure:

```
1:  find leftmost_P
2:  while($NOC <= MAX$) {
3:          do : {
4:                  find leftmost_P' (if more than two points have same $x$-axis value,
                    select upper point first)
5:                  compare x-dist and y-dist
6:                      if x-dist>y-dist
7:                          then x-dist_counter++
8:                      else y-dist_counter++
9:                  leftmost_P= leftmost_P'
10:                 $NOC++$
11:             }
12: }

13: if x-dist_counter > y-dist_counter
14:     then select HB for the broadcast data object
15: else select VB for the broadcast data object
```

The server decides the sequential order of the broadcast data objects based on the above algorithm. If the final value of the counter, *x-dist*, is bigger than that of *y-dist*, that is, the data objects are horizontally distributed, the server broadcasts the data objects in horizontal order. On the other hand, if the value of *y-dist* is bigger than that of *x-dist*, that is, the data objects are vertically distributed, the server broadcasts the data objects in vertical order. In this paper, we assume that the server broadcasts the data objects using HB.

In order to identify the desired data object using the BBS scheme, the client has to check whether or not the current broadcast data object is inside the query range during the *tuning time*. Let the client's current location be point q and the object's location be point p. Let point 'q' be the center of a circle from the query range and 'r' be the radius of this circle (see Fig. 5). From point 'q' if the point p is located inside the circle, then p is located inside the query range. The client may change the value of 'r', in order to adjust the size of the search scope. We denote the Euclidian distance between the two points p and q by $dist(p, q)$. In the map, we have $dist(p, q){:=}\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$.

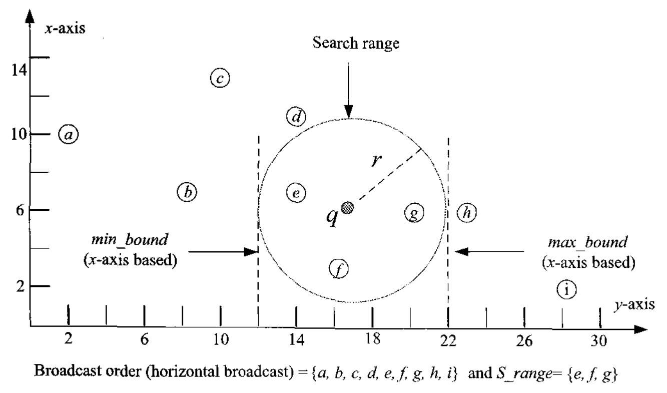

x-axis
Search range
min_bound (x-axis based)
max_bound (x-axis based)
y-axis
r
q
a
b
c
d
e
f
g
h
i
Broadcast order (horizontal broadcast) = {a, b, c, d, e, f, g, h, i} and S_range= {e, f, g}

Fig. 5. Detection of the data objects inside the search range.

The client uses the following algorithm to identify the desired data object.

Notations:

- S: Server data set
- O: Broadcast data object, where $O \in S$
- O_c: Current broadcast data object, where $O_c \in S$
- $O\_outside$: Data object that is located outside of the search range
- S_range: Set of data objects inside the search range, where $S\_range \in S$
- min_bound: Minimum boundary of the x-coordinate (when the server broadcasts data using HB) of the search range from the query point 'q' (see Fig. 5)
- max_bound: Maximum boundary of the x-coordinate (when the server broadcast data using HB) of the search range from the query point 'q'

Algorithm 2. The client algorithm used to identify the object inside the search range

Input: Locations of the clients and the data objects;
Output: Set of data objects inside the search range;
Procedure:

```
1:  if(current data object is an index segment)
2:      then find objects inside of the search range using an index segment
3:  else
4:      for each data object $O \in S$ {
5:          if($x$-coordinate of $O_c$ is smaller than $min\_bound$)
6:              then $O_c = O\_outside$
7:          else if($min\_bound \le x$-coordinate of $O_c \le max\_bound$)
            // when broadcast using HB
8:              then compare $dist(q, O_c)$ and $dist(q, r)$
9:                  if $dist(q, O_c) \le dist(q, r)$
10:                     then $O_c \in S\_range$
11:                 else $O_c = O\_outside$
12:         else
13:             stop tune and return the $S\_range$
14:     }
```

Lemma 1: If x-coordinate of $O_c < min\_bound$, then O_c is located outside of the search range.

Proof: Given a query point 'q' and the search range radius 'r', let O_c is an object 'b' in Fig. 5. Since the x-coordinate of O_c is outside of the circle, then O_c is located outside of the search range. □

Lemma 2: When the data objects are sequentially broadcasted in horizontal order, that is, from the leftmost coordinate to the rightmost coordinate, if the x-coordinate of $O_c >$ max_bound, then O_c and the rest of the broadcast data objects in the cycle are located outside of the search range.

Proof: Given a query point 'q' and the search range radius 'r', let O_c be an object 'h' (see Fig. 5). Since O_c and the rest of the broadcast objects' x-coordinates are greater than max_bound, object 'h' and 'i' are located outside of the search range. Thus, the client no longer needs to tune into the broadcast channel, and return $S\_range = \{e, f, g\}$. □

Lemma 3: The results of the Lemma 1 and the Lemma 2 lead us to the conclusion that if O_c satisfies the following steps, then $O_c \in S\_range$.

Step 1: The x-coordinate of O_c satisfies: $min\_bound \le x$-coordinate of $O_c \le max\_bound$

Step 2: $dist(q, O_c) \le dist(q, r)$

The following shows comparison of the *probe wait* and the *Bcast wait* between BBS and previous index method [3], [4]. Let m denote the number of times broadcast indices.

probe wait:

- Previous index method: $\frac{1}{2}(index + \frac{data}{m})$
- BBS method: None

Bcast wait:

- Previous index method: $\frac{1}{2}((m \times index) + data) + C$
- BBS method: $\frac{1}{2}((m \times index) + data) + C$

Since the *access latency* is the sum of the *probe wait* and the *Bcast wait*, average *access latency* for

- **Previous index method** is $\frac{1}{2}(index + \frac{data}{m}) + \frac{1}{2}((m \times index) + data) + C = \frac{1}{2}((m + 1)index + (\frac{1}{m} + 1)data) + C$.
- **BBS** is $\frac{1}{2}((m \times index) + data) + C$.

B. Sequential Prefetching and Caching Method

The result of the S_range may be changed if the client moves. Thus, the client has to tune its broadcast channel every time it moves. Data prefetching and caching has been proposed as a method of hiding the *access latency* of a data object. In this section, we present a sequential prefetching and caching method for use in LAMSs. In this method, the client prefetches and caches the data object according to the locations of the data objects for future use. Since the data objects are sequentially ordered, this scheme can significantly increase the cache hit ratio. Let w_c be the size of the cached data. The client adjusts the size of w_c according to the client's speed and the size of the cache. Let $P := \{p_{-n}, \cdots, p_{-2}, p_{-1}, p_0, p_1, p_2, \cdots, p_n\}$ be a set of n distinct points that represents the data objects, and q represents a query point.

Notations:

- $w, n \ge 0$ and $(w - n) \ge 0$
- NN: An object p_0, where $dist(p_0, q) \le dist(\forall p_{-n}, q)$ or $dist(p_0, q) \le dist(\forall p_n, q)$
- $p_{\min}$: An object p_{-w}, where $dist(p_{-(w-n)}, q) \le dist(p_{-w}, q) \le dist(p_{-(w+n)}, q)$
- $p_{\max}$: An object p_w, where $dist(p_{w-n}, q) \le dist(p_w, q) \le dist(p_{w+n}, q)$

Since the broadcast data objects are sorted sequentially based on their location, the client can easily identify the NN. After

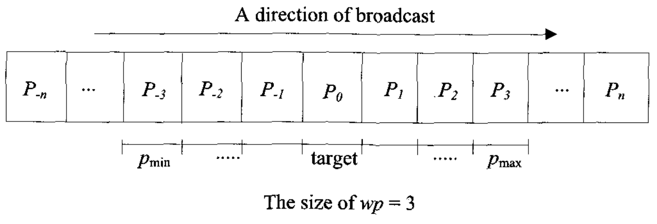


Fig. 6. An example of prefetching windows.

the client identifies the NN, it prefetches a data object from $p_{\min}$ to $p_{\max}$. We assume that the each client has a queue, in order to maintain previously broadcasted data objects. Let S_q be the set of data objects in the queue. Then, the number of returned data objects depends on the value of w_c. If we regard the value of w_c as n, the number of returned objects is $2n + 1$. Hence, if the value of n is 0, the number of returned objects is 1 (nearest neighbor), or if the value of n is 3, the number of returned objects is 7 (7-nearest neighbors).

Lemma 4: Given a point 'q', w_c contains $k - NN$ query set, if S_q is sorted according to the distance between the 'q' and p_i, where $p_i \in S_q$.

Proof: Let $S_q = \{p_i, p_{i+1}, p_{i+2}, \cdots, p_{i+n}\}$. If S_q is sorted ascending order according to the distance between the 'q' and p_i, then $dist(p_i, q) < dist(p_{i+1}, q) < dist(p_{i+2}, q) < \cdots < dist(p_{i+n}, q)$. Hence, if the value of $k - NN$ is n, set of $k - NN = \{p_i, p_{i+1}, p_{i+2}, \cdots, p_{i+n}\}$. Therefore, w_c contains $k - NN$ query set from a query point 'q'. □

The sequentially prefetched data objects are cached on the client side, and the client checks the validity of each cached data object using the following algorithm.

Notations:
- O^c: Object in the cache
- S^c_range: A set of cached data objects partially inside the range, where $S^c\_range \in S\_range \in S$

Algorithm 3. The client's validity checking algorithm for the cached data objects

input: Data objects in the cache;
output: Data objects inside the search range;
procedure:

```
1:  for each data $O_c \in S$ {
2:      if($x$-coordinate value of $O_c$ is smaller than $min\_bound$)
3:          then $O_c = O\_outside$ and marked as invalid
4:      else if($min\_bound \leq x$-coordinate of $O_c \leq max\_bound$)
        // when broadcast using HB
5:          then compare $dist(q, O_c)$ and $dist(q, r)$
6:              if $dist(q, O_c) \leq dist(q, r)$
7:                  then$O^c \in S^c\_range$ and marked as valid
8:              else $O^c = O\_outside$ and marked as invalid
9:      else
10:         $O^c = O\_outside$ and marked as invalid
11: }
12: return the $S^c\_range$
```

Lemma 5: If $O^c \in S^c\_range$, then $O^c \in S\_range$. Thus, O^c is inside the search range.

Proof: Let $S = \{a, b, c, d, e, f, g, h, i\}$, $S\_range = \{e, f, g\}$, and $S^c\_range = \{q\}$ (see the Fig. 5). If we assume that the object 'q' is cached on the client side and $q \in S^c\_range$, then the object 'q' is inside the search range, since $S^c\_range \in S\_range \in S$. □

Table 1. Simulation parameters.

Description	Setting
Service area	1000×1000 (km^2)
% of service area	30–100
No. data objects	10–1000
Size of data object	256–8192 bytes
Broadcast bandwidth	144 kbps
No. of clients	0–90
Minimum moving speed of the client	10 (km/h)
Maximum moving speed of the client	90 (km/h)
Distance of move	5–50
size of w_c	0–5
No. of broadcast period	50–100
Size of max_OBC	longer than 900 m

V. PERFORMANCE EVALUATION

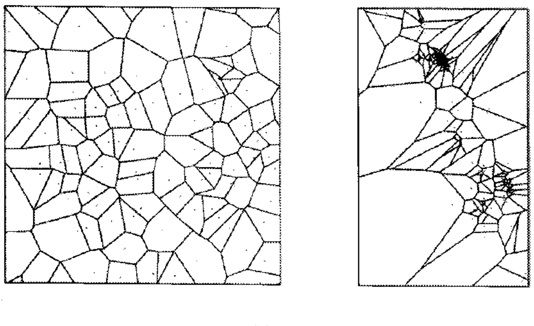
In this paper, we evaluate the performance with various kinds of parameters settings such as the client's speed, the size of the service area, and the distributions of the data objects. Then, we evaluate the cache hit ratio with parameters such as the size of w_c and the service coverage area. Finally, we compare the performance of the BBS scheme, the R-tree index [17] scheme, and the D-tree index [2] scheme. We assume that the broadcast data objects are static such as restaurants, hospitals, and hotels. We use a system model similar to that in [13] and [18]. The whole geometric service area is divided into groups of mobile supporting stations (MSSs). In this paper, two datasets are used in the experiments (see Fig. 7(a)). The first data set $\mathcal{D}1$ contains data objects randomly distributed in a square Euclidian space. The second data set $\mathcal{D}2$ contains the data objects of hospitals in the Southern California area, which is extracted from the data set at [19]. Table 1 shows the notations and default parameter settings used in the simulation.

A. Latency

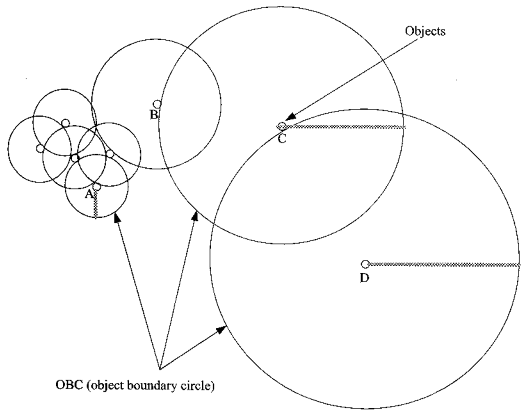
In this section, we evaluate the *access latency* for various parameter settings, such as the client's speed, the size of the service area, and the number of clients. In this paper, we present the object boundary circle (OBC) which represents the distance between the objects, as shown in Fig. 7(b). The radius of this circle represents the distance between the objects, and the circle with the longest radius is selected as a hot data object such as C and D in Fig. 7(b). The server broadcasts data objects with different frequencies, such as hot and cold data objects [14]. The data objects that are selected as a hot data is broadcasted more frequent than others in a single broadcast cycle.

A.1 Effect of the Size of the Service Area

In this section, we study the effect of the size of the service area according to the client's speed. We vary the service coverage area from 5% to 100% of the whole geographic area. The query arrival time decreases as the size of the service area decreases, since the size of the entire broadcast is reduced. However, the query arrival time is significantly increased when the client's speed increase and the client goes outside of the ser-



(a)



(b)

Fig. 7. Scope distributions and OBC: (a)Two scope distributions for performance evaluation, (b) OBC; (1) Min_OBC: Minimum boundary circle is picked as a hot data object, such as object A (2) max_OBC: Maximum boundary circle is picked as a hot data object, such as object D (3) uniform: All data objects are broadcasted in same frequency.

vice coverage area, as shown Fig. 8(a). In this case, the client's cached data items become invalid and the client has to tune the broadcast channel again.

A.2 Effect of the Client's Speed

In this section, we study the effect of the client speed. First, we vary the client's speed from 5 to 50 in $\mathcal{D}1$. When the client's speed is the lowest, broadcast size of 10% (of the coverage area) is the best. However, as the client's speed increases, its performance is degraded in comparison with that of the other clients, since for most values of the client's speed, the client is outside of the service coverage area, as shown in Fig. 8(b). Second, we study the performance for different parameters such as min_OBC, max_OBC, and uniform (see Fig. 7(b)) in $\mathcal{D}2$. In this experiment, we assume that the clients are uniformly distributed in the map. Fig. 8(c) shows the result as the client speed increases from 5 to 50, and Fig. 8(d) shows the result as the number of clients increased from 15 to 90. As shown in figure, max_OBC outperforms compare to others.

A.3 Effect of the Distribution of Data Objects and the Clients' Location

In this section, we study the effect of the distributions of the data objects and the clients' location. First, we assume that

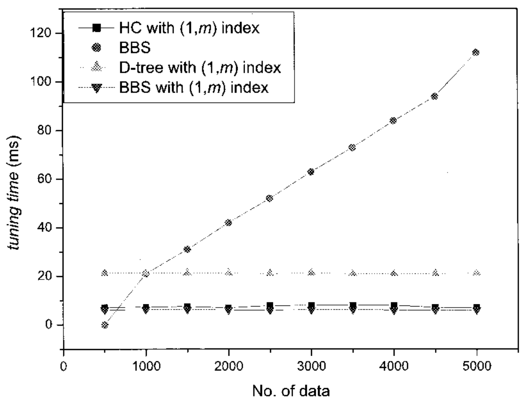


Fig. 9. Compare the *tuning time* of BBS with D-tree and Hilbert-curve index.

the clients are crowded in a specific region such as downtown. Those data objects which are located in such a region are selected as hot data objects. In this experiment, we evaluate the performance in relation to four different parameters as shown in Table 2.

Fig. 8(e) shows the result as the number of client is increased from 15 to 90 in $\mathcal{D}1$. As shown in figure, the $hot\_50\%$ outperforms compare to others as the number of client increases. Second, we assume that the clients are uniformly distributed in $\mathcal{D}2$. Fig. 8(f) shows the result as the number of client increases from 15 to 90. As shown in figure, in this case, the broadcast hot data object does not affect the query response time since the clients are uniformly distributed in the map. However, the size of the service area affect the query response time.

B. Comparison of the Tuning Time of the BBS with the D-Tree and Hilbert-Curve Index

In this section, we we compare the BBS scheme with the D-tree and the Hilbert-curve index schemes. Let HC denote the Hilbert-curve index. In BBS, no index is interleaved in a broadcast cycle with the data items. On the other hand, BBS with $(1, m)$, D-tree with $(1, m)$, and HC with $(1, m)$ methods interleave an index m times in a broadcast cycle with the data items. As shown in the Fig. 9, the BBS with $(1, m)$ index outperforms compare to others, since the sequential access property of the wireless broadcast. As shown in the figure, *tuning time* of the BBS significantly increases compare to others as the number of data increases. The reason is the client tune into the broadcast channel and filter all data till the required data items are downloaded.

C. Comparison of the Performance of the BBS Scheme with the D-Tree and the Hilber-Curve Index

In this section, we compare the BBS scheme with the D-tree and the D-tree index schemes. First, we vary the number of the data objects from 100 bytes to 500 in $\mathcal{D}2$. Since the clients do not need to wait and tune into the broadcast channel to receive an index segment, if they have already identified the desired objects, the BBS scheme shows increasingly lower latency compared to the D-tree and the Hilbert-curve index scheme. Fig. 10(a) shows the result as the number of data increases.

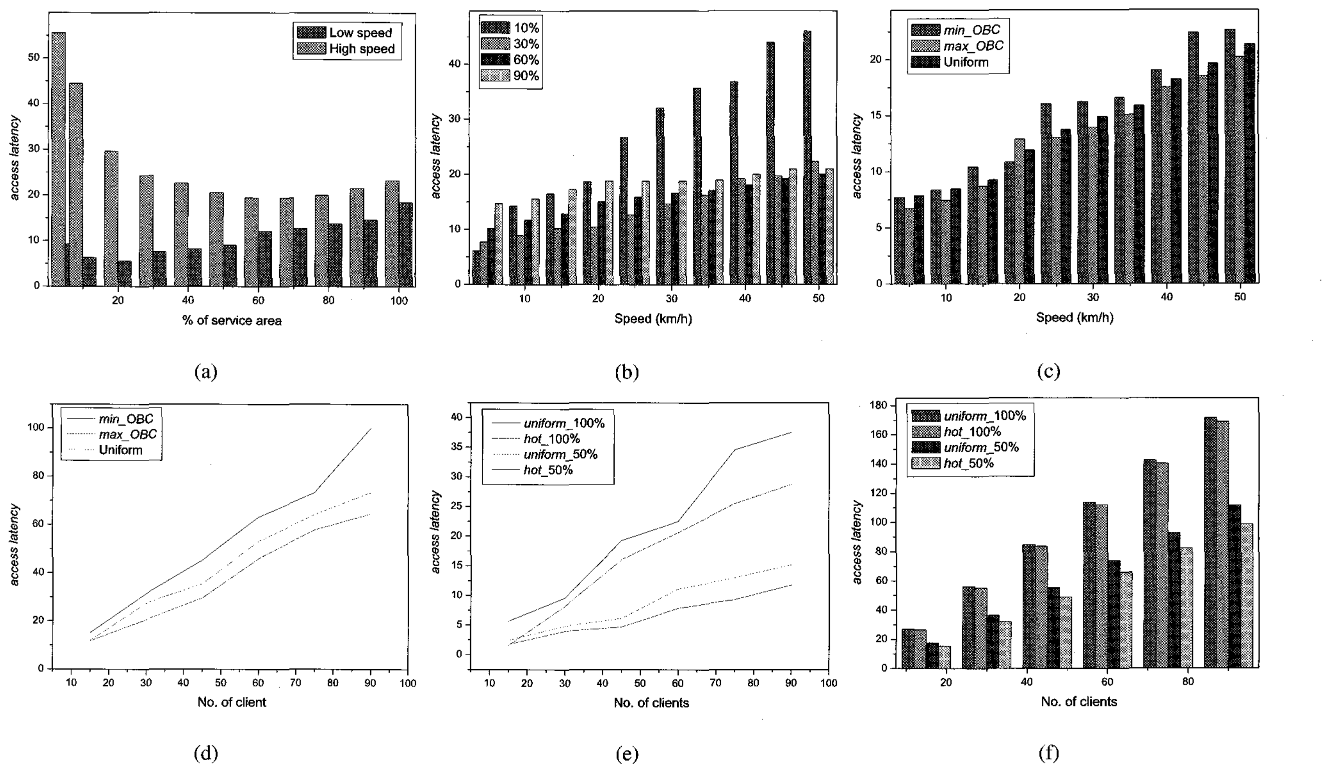


Fig. 8. *access latency*: (a) *access latency* as the % of servece area increase, (b) *access latency* as the client speed increase, (c) *access latency* as the client speed increase, (d) *access latency* as the number of client increase, (e) *access latency* as the number of client increase, (f) *access latency* as the number of client increase.

Table 2. Broadcast environment setting.

Description	Setting
$uniform\_100\%$	The server broadcasts data objects with the same frequency such as flat broadcast in [14] and the service coverage area is the whole geographic area.
$hot\_100\%$	The server broadcasts data objects with different frequencies such as those corresponding to hot and cold data objects and the service coverage area is the whole geographic area.
$uniform\_50\%$	The server broadcasts data objects with the same frequency and the service coverage area is set to 50% of the whole geographic area.
$hot\_50\%$	The server broadcasts data objects with different frequencies such as those corresponding to hot and cold data objects and the service coverage area is set to 50% of the whole geographic area.

Then, we vary the number of clients from 50 to 300. Fig. 10(b) shows the result as the number of client increases. In this experiment, the server periodically broadcasts 506 data objects to the clients. Figs. 10(c) and 10(d) show the results of the *access latency* as the client's moving distance and the size of the search range increases, respectively.

D. Cache Hit Ratio

This section evaluates the cache hit ratio for various parameters settings such as the size of w_c, the client's speed and the size of the service area. First, we vary the client's speed from 10 to 50 in $\mathcal{D}2$. As shown in Fig. 11(a), the number of cache hits decreases as the client's speed is increased. The broadcast hot data object does not affect the client's cache hit ratio. In this case, the $uniform\_100\%$ outperforms the $uniform\_50\%$ since clients discard the cached data object if they move to the other service area. Second, we vary the client's speed from 10 to 50 in $\mathcal{D}1$. As shown in Fig. 11(b), the number of cache hits decreases as the client's speed is increased. Third, we vary the value of w_c from 1 to 5 in $\mathcal{D}1$. As shown in Fig. 11(c), the number of cache hits increases as the client's speed is decreased and the size of w_c increases.

VI. CONCLUSION

In this paper, we studied the different broadcasting and prefetching schemes which can be used for LAMSs. For the purpose of broadcasting in LAMSs, we present the BBS and sequential prefetching and caching methods. In these methods, we do not modify the previous index scheme. Rather, we sim-

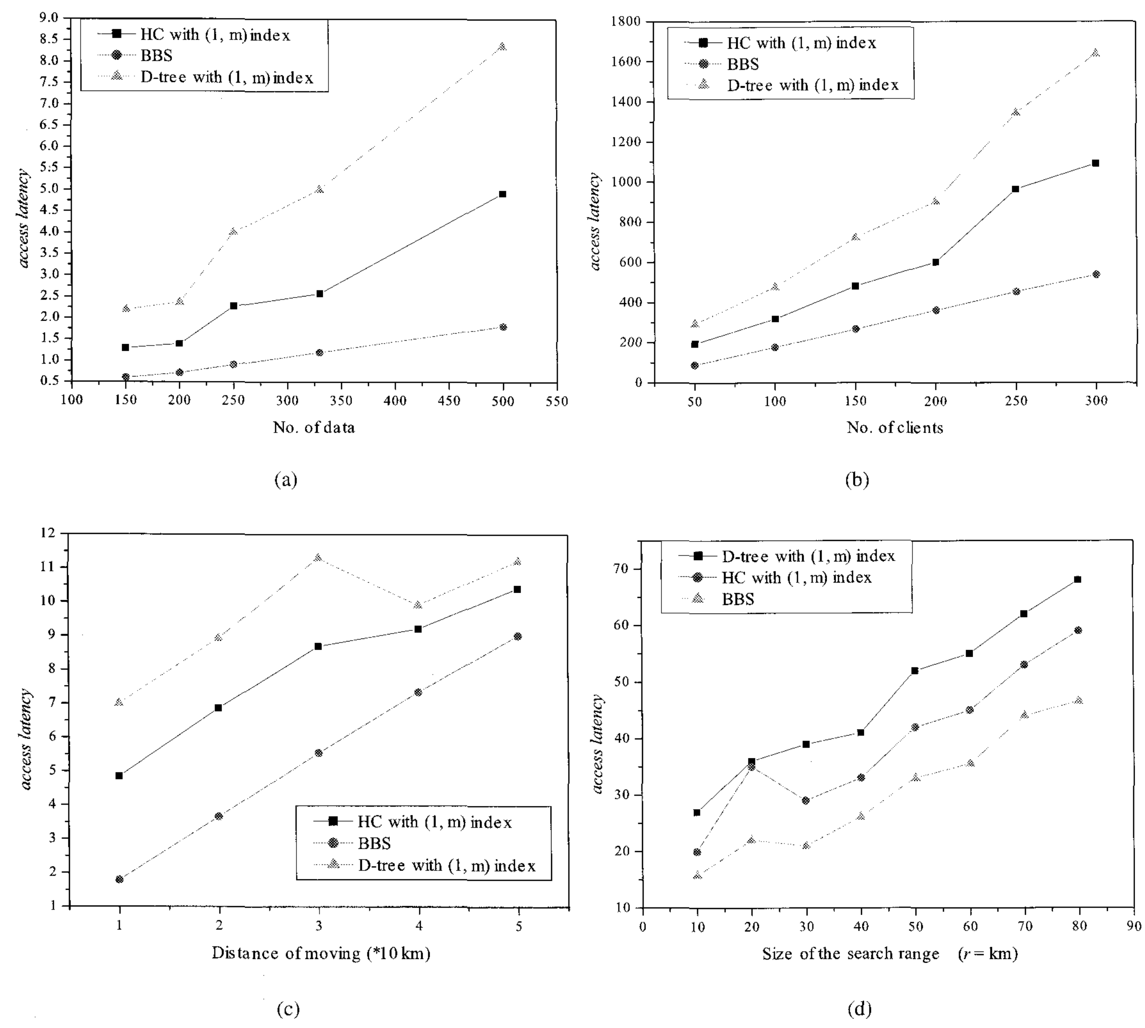


Fig. 10. Compare the *access latency* of BBS Scheme with the R-tree and the D-tree Index: (a) *access latency* as the number of client increase, (b) *access latency* as the number of client increase, (c) *access latency* as the client's moving distance increase, (d) *access latency* as the size of the search range increase.

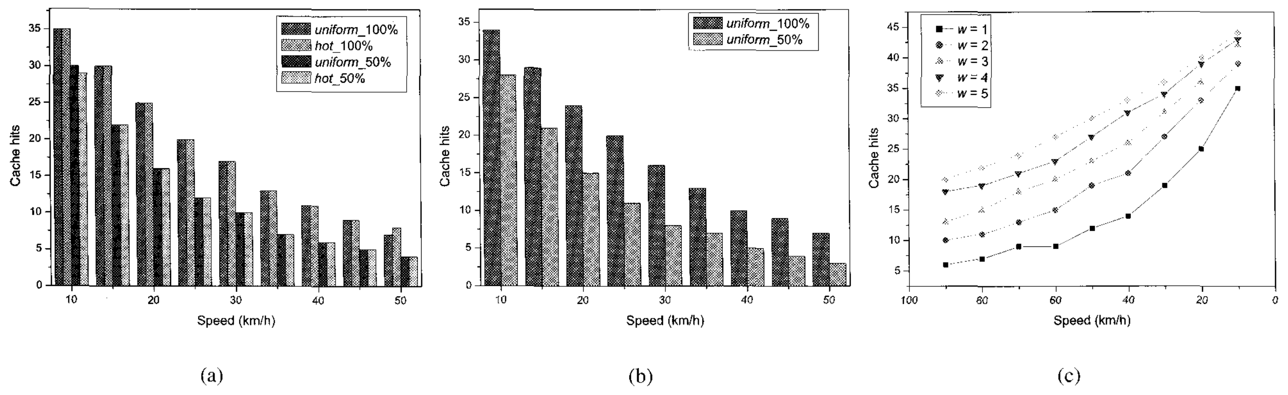


Fig. 11. Cache hit ratio: (a) Cache hits as the client's speed increase in $\mathcal{D}2$, (b) cache hits as the client's speed increase in $\mathcal{D}1$, (c) cache hits as the client's speed decrease.

ply sort the data objects based on their locations, and the server then broadcasts them sequentially to the mobile clients. The BBS method attempts to reduce the *access latency* for the client. Furthermore, the proposed sequential prefetching and caching method can also reduce the query response time and *tuning time*, respectively. With the proposed schemes, the client can perform range query processing without having to tune into the broadcast channel, in the case where the desired data objects have already been identified or prefetched into the cache.

The proposed schemes were investigated in relation to various environmental variables, such as the distributions of the data objects, the average speed of the client and the size of the service area. The experimental results show that the proposed BBS scheme significantly reduces the *access latency* compared with the R-tree index and D-tree schemes, since the client does not always have to wait for the index segment.

In this paper, we did not consider the case of moving data objects in LAMSs. Hence, we are planning to extend this study to the case of a moving object database. Finally, we are also planning to investigate the issue of cache replacement in a future study.

REFERENCES

[1] K. Park, M. Song, and C. Hwang, "An efficient data dissemination schemes for location dependent information services," *Lecture Notes in Computer Science 3347*, Springer-Verlag, 2004, pp. 96–105.

[2] J. Xu, B. Zheng, W.-C. Lee, and D. L. Lee, "D-tree: An index structure for planar point queries in location-based wireless services," *IEEE Trans. Knowledge Data Eng.*, 2004.

[3] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Energy efficient indexing on air," in *Proc. SIGMOD'94*, 1994.

[4] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on air: Organization and access," *IEEE Trans. Knowledge Data Eng.*, vol. 9, no. 3, 1997.

[5] D. L. Lee, J. Xu, and B. Zheng, "Data management in location-dependent information services," *IEEE Pervasive Computing*, vol. 1, no. 3, 2002.

[6] W.-C. Lee and B. Zheng, "A fully distributed spatial index for wireless data broadcast," in *Proc. ICDE 2005*, 2005.

[7] B. Gedik, A. Singh, L. Liu, "Energy efficient exact KNN search in wireless broadcast environments," in *Proc. ACM GIS 2004*, 2004.

[8] B. Zheng, W.-C. Lee, and D. L. Lee, "Spatial queries in wireless broadcast systems," *Wireless Network*, 2004.

[9] B. Zheng, W.-C. Lee, and D. Lee, "Spatial index on air," in *Proc. PerCom 2004*, 2004.

[10] B. Zheng, W.-C. Lee, and D. Lee, "Search K-nearest neighbors on air," in *Proc. MDM 2003*, 2003.

[11] S.E. Hambrusch, C.-M. Liu, W. Aref, and S. Prabhakar, "Query processing in broadcasted spatial index trees," in *Proc. SSTD 2001*, 2001.

[12] J. Xu, X. Tang, and D. L. Lee, "Performance analysis of location-dependent cache invalidation schemes for mobile environments," *IEEE Trans. Knowledge Data Eng.*, vol. 2, no. 15, 2003.

[13] B. Zheng, J. Xu, and D. L. Lee, "Cache invalidation and replacement strategies for location-dependent data in mobile environments," *IEEE Trans. Compt.*, vol. 51, no. 10, 2002.

[14] S. Acharya and M. Franklin, "Broadcast disks: Data management for asymmetric communication environments," in *Proc. SIGMOD'95*, 1995.

[15] W. C. Lee and D. L. Lee, "Using signature techniques for information filtering in wireless and mobile environments," *Distributed and Parallel Databases*, vol. 4, no. 3, 1996.

[16] Q. L. Hu, W.-C. Lee, and D. L. Lee, "A hybrid index technique for power efficient data broadcast," *Distributed and Parallel Databases*, vol. 9, no. 2, 2001.

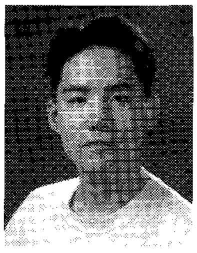
[17] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. SIGMOD'84*, 1984.

[18] D. Barbara and T. Imielinski, "Sleepers and workaholics: Cashing strategies in mobile environments," in *Proc. SIGMOD'94*, 1994.

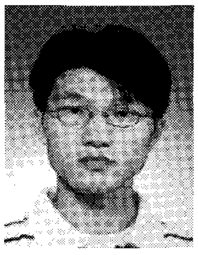
[19] Spatial Datasets, available at http://dias.cti.gr/ytheod/research/datasets/spatial.html.

[20] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proc. SIGMOD'95*, 1995.

[21] J. Xu, B. Zheng, W.-C. Lee, and D. L. Lee, "Energy efficient index for querying location-dependent data in mobile broadcast environments," in *Proc. ICDE 2003*, 2003.



Kwangjin Park received his B.S. and M.S. degrees in Computer Science from Korea University, Korea in 2000 and 2002, respectively. He is currently a Ph.D. candidate in Computer Science and Engineering and a researcher in the Research Institute of Computer Information and Communication, Korea University, Korea. His research interests include location-dependent information systems, mobile databases, and mobile computing systems.



Moonbae Song received his B.S. degree in computer science from Kunsan National University in 1996, and his M.S. degree in computer science from Soongsil University in 1998. He received his Ph.D. in computer science from Korea University in 2005. Currently, he is a research assistant professor in the Research Institute of Computer Information and Communication at Korea University. His research interests include moving object databases, location-aware services, context-awareness, and data management for mobile computing.

Chong-Sun Hwang received his M.S. degree in Mathematics from Korea University, Seoul, Korea in 1970, and his Ph.D. degree in Statistics and Computer Science from the University of Georgia in 1978. From 1978 to 1980, he was an assistant professor at South Carolina Lander State University. He is currently a full professor in the Department of Computer Science and Engineering at Korea University, Seoul, Korea. Since 1995, he has been a Dean in the Graduate School of Computer Science and Technology at Korea University. His recent research interests include distributed systems, distributed algorithms, and mobile computing systems.