An Efficient Data Dissemination Scheme for Nearest Neighbour Query Processing

KwangJin Park
Department of Computer Science and Engineering, Korea University
5-1, Anam-dong, Seongbuk-Ku, Seoul 136-701, Korea
kjpark@disys.korea.ac.kr

Location dependent information services (LDISs) produce answers to queries according to the location of the client issuing the query. In LDIS, techniques such as caching, prefetching and broadcasting are effective approaches to reducing the wireless bandwidth requirement and query response time. However, the client's mobility may lead to inconsistency problems. In this paper, we introduce the broadcast-based LDIS scheme (BBS) for the mobile computing environment. In the BBS, broadcasted data items are sorted sequentially based on their location and the server broadcasts the location dependent data (LDD) along with an index segment. Then, we present a data prefetching scheme and OBC (Object Boundary Circle), in order to reduce the query response time. The performance of the proposed scheme is investigated in relation to various environmental variables, such as the distributions of the data items, the average speed of the clients and the size of the service area.

ACM Classification: H.2.8

1. INTRODUCTION

Mobile location-dependent information services (LDISs) have drawn a lot of attention from wireless data industries in the past few years. This growth in mobile communications presents new aspects to the resource assignment problem as well as new applications. In these services, information provided to mobile users' reflects their current geographical locations. Location dependent data is a data whose value depends on the location. The answer to a query depends on the geographical location where the query originates. Let's consider an example in which a user drives a car and wants to find the nearest gas stations. The user sends a query, such as, “what are the names and locations of the gas stations near to my current location?”, using his mobile device. Once the user gets the answer from the server, he will visit the gas stations in order of the nearest to his location based on price. To handle such a query, the positions of the objects and the user must be found. In this paper, we propose the broadcast-based LDIS scheme under a geometric location model. We first introduce the broadcast based location dependent data delivery scheme (BBS). In this scheme, the server periodically broadcasts reports, which contains the IDs of the data items (e.g., building names) and the values of the location coordinates to the clients. The broadcasted data objects are sorted sequentially, based on their location before being broadcasted. Then, we introduce the prefetching scheme in LDIS for the mobile computing environment. By using the proposed
schemes, the client’s access and tuning times are significantly reduced. The main contributions of our work can be summarized as follows:

- It is not necessary for the client to wait for and tune into a particular index segment, if it has already identified the nearest object before the index segment has arrived. This technique significantly reduces the access time in the broadcast based LDIS environment.
- The client simply adjusts the value of k when it performs the k-NN query processing.
- The client can also perform the a k-NN query processing without an index segment. In this case, the best access time is obtained, since no index is broadcast along with the file (Imielinski et al, 1994).

The rest of the paper is organized as follows: Section 2 gives the background of the broadcast model and LDIS scheme. Section 3 describes the proposed algorithms. The performance analysis and evaluation is presented in Sections 4 and 5 respectively. Finally, Section 6 concludes this paper.

2. 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 LDISs (Lee et al, 2002), 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.

2.1 Broadcast Model
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. (1,m) index is one of techniques that attempts to address this issue, by interleaving indexing information among the broadcast data items (Imielinski et al, 1997; Imielinski et al, 1994). At the same time, the client can reduce its battery power consumption through the use of select tuning. By accessing the index segment, the mobile client is able to predict the arrival time of the desired data item. Thus, it can stay in the power saving mode most of the time, and tune into the broadcast channel only when the requested data arrives. (1,m)index techniques can be evaluated in terms of the following factors:

- **Access Time**: 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.
- **Tuning Time**: The amount of time spent by a client listening to the channel.
- **The Access Time**: 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 segments 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 Time** is the sum of the **Probe Wait** and **Bcast Wait**. These two factors work against each other (Imielinski et al, 1997; Imielinski et al, 1994). There are several indexing techniques such as the distributed indexing approach (Imielinski et al, 1994), the signature approach (Lee and Lee, 1996), and the hybrid approach (Hu et al, 2001).
2.2 Nearest Neighbour Searching
The most popular class of problems in LDIs is nearest-neighbour (NN) search. An example of a NN search is: “Show me the nearest restaurant.” A lot of research has been carried out on how to solve the NN search problem for spatial databases. In Roussopoulos et al. (1995), authors propose a branch-and-bound R-tree traversal algorithm to find the nearest neighbour object to a point, and then generalize it to finding the K-nearest neighbours. In Zheng et al. (2003), authors address the issues involved with organizing location dependent data and answering K-nearest neighbour queries on air.

Definition 2.1: Given a query point q, find the spatial data object O with the smallest distance to q.

\[ \text{NN}(q) = \{O \mid \text{dist}(O, q) \leq \text{dist}(O', q)\} \] (1)

2.3 LDIs Schemes
In the mobile computing environment, caching data at the client’s side is a useful technique for improving the performance. However, the frequency of disconnection and the mobility of the clients may cause cache inconsistency problems. In Hue et al. (2001), 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 is 00010011, this means that the data item is valid in 4th, 7th, and 8th only. Then, they organize each service area as a group in order to reduce the overhead for scope information. In Zheng et al. (2002), authors proposed a PE (Polygonal Endpoint) and AC (Approximate Circle) 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.

3. MOTIVATION
Mobile clients in wireless environments suffer from scarce bandwidth, low-quality communication, frequent network disconnection, and limited local resources. Data prefetching and caching on mobile clients have been considered an effective solution to improve system performance (Hu et al., 2001; Zheng et al., 2002). The idea is to transfer information, which the user might need in the future, so that it is already stored on the users mobile device when it is accessed. This technique has two major advantages. First, the system has lower response times because more data is available in the cache. Second, there is less “burst” load placed on the network because prefetching is done only when there is sufficient bandwidth available rather than on demand (Kirchner et al., 2004). Then, the client stores information on the cache so that queries can be answered without connecting to the server.

Let’s consider an example in which the user is driving on the highway from Region 1 to Region 3 and desires to obtain the nearest gas station as shown in Figure 1. When the user moves within Region 1, Region 2 or Region 3, the nearest object is G1, G2 or G3 respectively. That is, if the user is located inside Region n, any query to find the nearest gas station will return the same gas station.

Let us assume that the server broadcasts three data items, such as data objects 1, 2 and 3 (i.e., gas station G1, G2 and G3). First, the client obtains G1 and stores it in the cache via the wireless data broadcast channel. Since the client has G1 in the cache, it is not necessary for the client to tune the broadcast channel again while it is located at Region 1. Moreover, even if the user’s location has
changed within Region 1 and they repeatedly request such information, it is not necessary for the client to tune the broadcast channel. Instead, the client answers the user’s requests by using the cached data items. Then, the client prefetches G2 and G3. Since the client has G2 and G3 in the cache, it is not necessary for the client to tune the broadcast channel again while it moves from Region 2 and Region 3.

This example tells us the importance of the data prefetching and the caching technique for spatial query processing. Moreover, the query response time is greatly affected in the order at which the data items are broadcast. Therefore, the broadcast sequence must be considered in the localities of the data objects for location-dependent spatial queries.

4. PROPOSED ALGORITHMS

In this section, we present the concept of the location-based spatial data dissemination and describe the processing of mobile k-NN search. We assume that a mobile client knows its position, say through GPS.

4.1 Location-based Spatial Data Dissemination

In response to the linear characteristics of wireless broadcast, the spatial data dissemination method for LDISs by linear scanning of the data set is proposed, namely BBS. In this method, the server broadcasts data objects that are sorted sequentially according to the location of the data objects. Moreover, based on the distance between the data objects, we assign the different weight values to each data object, by using the OBC (Object Boundary Circle). Also, the data objects can be sent using different broadcast frequencies, by classifying them into hot and cold groups (Acharya and Franklin, 1995). For instance, the data objects selected as a hot group will broadcast more frequently than the other groups. We discuss this issue in the section concerning the performance evaluation. In the BBS method, since the data objects broadcast 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 nearest object before the associated index segment has arrived. We assume each data item contains the index pointer header information that represents the offset to the beginning of the next index segment, such as Imielinski et al (1997) and Imielinski et al (1994).

Figure 1: Prefetching and Caching for the Spatial Data
In this method, the structure of the broadcast affects the distribution of the data objects. For example, as shown in Figure 2, if the data objects are horizontally distributed, the server broadcasts data objects sequentially, from the leftmost data object to the rightmost data object. A simple sequential broadcast can be generated by linearizing the two dimension coordinates in two different ways: i.e., horizontal broadcasting (HB) or vertical broadcasting (VB). In HB, the server broadcasts the LDD in horizontal order, that is, from the leftmost coordinate to the rightmost coordinate. 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 HB or VB, the server uses the following algorithm:

**Notations:**
- $S$: server data set
- $S_c$: compare data set, where $S_c \subseteq S$ and initially $S_c = S$
- $\text{leftmost}_P$: a point that is located at the leftmost extremity in the map (e.g., object ‘a’ in Figure 2)
- $\text{leftmost}_P'$: a point that is located at the leftmost extremity in the map with the exception of $\text{leftmost}_P$, where the value of the coordinates of $\text{leftmost}_P'$ is leftmost $P$ (e.g., object ‘b’ in Figure 2)
- $x$-dist: distance between $\text{leftmost}_P$ and $\text{leftmost}_P'$ based on x-axis
- $y$-dist: distance between $\text{leftmost}_P$ and $\text{leftmost}_P'$ based on y-axis
- $x$-dist$_\text{counter}$: initial value is 0
- $y$-dist$_\text{counter}$: 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. Sort the data items $\in S$ in ascending order based on their x-coordinates
2. for (every data items $\in S$) {
3. do {
4. find $\text{leftmost}_P$ and $\text{leftmost}_P'$ from $S_c$ (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 increase $x$-dist$_\text{counter}$
8. else increase $y$-dist$_\text{counter}$
9. remove $\text{leftmost}_P$ from $S_c$
10. }
11. }
12. if $x$-dist$_\text{counter}$ $>$ $y$-dist$_\text{counter}$
13. then select HB for the broadcast data object
14. else select VB for the broadcast data object

In order to identify the nearest object using the BBS scheme, the client has to compare the most recently delivered data object with the previous one during the tuning time. The client uses the following algorithm to identify the nearest object:
An Efficient Data Dissemination Scheme for Nearest Neighbour Query Processing

Notations:
- \( O \): broadcast data object, where \( O \in S \)
- \( O_{candi} \): candidate for the nearest data object
- \( NN \): nearest neighbour data object
- \( O_c \): current broadcast data object (initially \( O_c \) regarded as NN), where \( O_c \in S \)
- \( O_p \): previous broadcast data object, where \( O_p \in S \)
- \( C_l \): client’s location

Algorithm 2. The client algorithm used to identify the nearest object

Input: locations of the clients and the data objects;
Output: NN;
Procedure:

1: if (current data item is an index segment)
2: then find NN using an index segment
3: else
4: do
5: for each data object \( O \in S \) {
6: if \( (O_c \) is the first broadcast data object)
7: then \( O_c = O_{candi} \)
8: else if \( (dist(O_c, C_l) < dist(O_p, C_l)) \)
9: then \( O_c = O_{candi} \)
10: else \( O_p = O_{candi} \)
11: }while (getting to the index segment or \( dist((x\text{-coordinate value of } O_c),(x\text{-coordinate value of } C_l)) < dist(O_{candi}, C_l)) \)
12: \( O_{candi} = NN \)
13: return NN

Lemma 4.1: While the data objects are sequentially broadcast in horizontal order, that is, from the leftmost coordinate to the rightmost coordinate, if \( O_c = O_i \), where \( O_i \in S \) and \( dist((x\text{-coordinate value of } O_i),(x\text{-coordinate value of } C_l)) > dist(O_{candi}, C_l) \), then \( O_i \) and the rest of the broadcast data objects are located outside of the NN range.

Proof: Given a query point ‘q’, if the \( O_{candi} \) is an object ‘e’, as shown in Figure 1. If \( dist((x\text{-coordinate value of the object ‘f’}),(x\text{-coordinate value of ‘q’})) > dist(‘e’,‘q’), \), then the objects ‘f’ and ‘g’ are located outside of the NN range and thus the client stops tuning the broadcast channel and selects object ‘e’ as the NN.

Since it does not have the location information of all of the data objects, the client cannot estimate which data will be broadcast next. Hence, even if the server delivers data objects sequentially based on their coordinate values, it is difficult to determine which data object is the nearest to the client. If the client loses the desired data object, it has to wait until the next broadcast period.

In our scheme, the client maintains a queue and determines the size of window \( w \) (hereafter referred to as \( w_q \)), which indicates the number of data objects that will be left in the queue. The client maintains objects in the queue based on the size of \( w_q \) and this queue can be represented as follows:
An Efficient Data Dissemination Scheme for Nearest Neighbour Query Processing

**Notations:**
- $O_j$: an object in the map
- $T_o$: the timestamp of an object
- $T_c$: the timestamp of the current broadcasted object
- $S$: set of objects in the map
- $S_q$: set of objects in the queue
- $w_q$: size of the windows in the queue

Then $S_q = \{<O_j, T_o>| (O_j \in S) \land (T_c - w_q \leq T_o \leq T_c)\}$

The following shows comparison of the Probe Wait and the Bcast Wait between BBS and previous index method (Imielinski et al., 1997; Imielinski et al., 1994). Let $m$ denote the number of times broadcast indices:

**Probe Wait:**
Previous index method: $\frac{1}{2} \times (\text{index} + \frac{\text{data}}{m})$

BBS method: None

**Bcast Wait:**
Previous index method: $\frac{1}{2} \times ((m \times \text{index}) + \text{data})
An Efficient Data Dissemination Scheme for Nearest Neighbour Query Processing

BBS method: \[ \frac{1}{2} \times (m \times \text{index} + \text{data}) \]

Since the Access Time is the sum of the Probe Wait and the Bcast Wait, average access time for previous index method is:

\[ \frac{1}{2} \times (\text{index} + \frac{\text{data}}{m}) + \frac{1}{2} \times ((m \times \text{index}) + \text{data}) = \frac{1}{2} \times ((m+1) \times \text{index} + (\frac{1}{m} + 1) \times \text{data}) \]  

(2)

and BBS is:

\[ \frac{1}{2} \times ((m \times \text{index}) + \text{data}) \]

(3)

**Definition 4.1:** While the data objects are sequentially broadcast in horizontal order, that is, from the leftmost coordinate to the rightmost coordinate, if x-coordinate of \( O_c \) > x-coordinate of \( q \), then the client stops tuning to the broadcast channel and switches to doze mode (power saving mode), until the next index segment arrives.

### 4.2 k-NN Queries via Wireless Data Broadcast

To avoid excessive network traffic and processor cycle use, the mechanism has to consider different strategies to increase the efficiency of the algorithm and the relevance of the fetched data. A powerful tool to address the prefetching problem in the context of mobile clients is location-awareness (Kirchner et al., 2004). The location-aware prefetching mechanism seeks to optimize the use of limited bandwidth networks and to provide users with low-latency information provisioning and updates, even in bad infrastructure areas.

In this section, we present a prefetching method for use in LDIS. In this method, the client prefetches the data object for future use. Let \( w_p \) be the size of prefetched data objects. The client adjusts the size of \( w_p \) according to the speed and size of the cache. Moreover, in order to adjust the value of \( k \) based on the \( k \)-nearest objects, the proposed scheme simply adjusts the size of \( w_p \). Let client’s current location be point \( q \) and object’s location be point \( p \). Then, we denote the Euclidean distance between the two points \( p \) and \( q \) by \( \text{dist}(p, q) \). Let \( P := \{p_{-n}, p_{-n+1}, p_0, p_1, p_2, \ldots, p_n\} \) be a set of \( n \) distinct points that represent the data objects, and \( q \) represents a query point.

**Notations:**
- \( \text{dist}(p_{-n}, q) \leq \text{dist}(p_{-n+1}, q) \leq \text{dist}(p_0, q) \)
- \( \text{dist}(p_{w-n}, q) \leq \text{dist}(p_w, q) \leq \text{dist}(p_{w+n}, q) \)
- \( \text{dist}(p_{-w}, q) \leq \text{dist}(p_0, q) \leq \text{dist}(p_{w}, q) \)

The client stores the prefetched data items in the cache for future use. Let \( w_p \) be the size of windows for prefetching data items. Then, the number of returned objects depends on the value of \( w_p \). If we regard the value of \( w_p \) as \( n \), the number of returned objects is \( 2n+1 \). In order to adjust the value of \( k \) of the \( k \)-nearest objects, the proposed scheme simply adjusts the size of \( w_p \). The formal description of the algorithm used for prefetching at the client side is as follows:

**Algorithm 3. Client algorithm for data prefetching**

**input:** sorted broadcast data objects according to the distance between the \( q \) and the data object;
output: set of final k-NN;

procedure:
1: while (a client looking for the nearest object) {
2: active mode (listen to the broadcast channel)
3: if (desired data comes from the server) { // by using algorithm 1
4: then current broadcast data object= \( p_0 \) and prefetch a data object from \( p_{\min} \) to \( p_{\max} \})
5: else
6: wait until the desired data comes from the server
7: }
8: doze mode

Lemma 4.2: Given a point ‘q’, \( w_p \) 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 \( S_q = \{p_1, p_{i+1}, p_{i+2} \ldots p_{i+n}\} \). If \( S_q \) is sorted ascending order according to the distance between the ‘q’ and \( p_i \), then \( \text{dist}(p_i, q) < \text{dist}(p_{i+1}, q) < \text{dist}(p_{i+2}, q) \ldots < \text{dist}(p_{i+n}, q) \). Therefore, \( w_p \) contains k-NN query set from a query point ‘q’.

5. PERFORMANCE EVALUATION
In this paper, we evaluate the performance with various kinds of parameter 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 \( w_q \) and the service coverage area. Finally, we compare the performance of the BBS scheme and the R-Tree index (Gutman, 1984) scheme and the D-tree index (Xu et al., 2004) 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 Zheng et al (2002) and Barbara (1994). The distance can be computed using the Euclidian distance between the two points \( p \) and \( q \) as defined by \( \text{dist}(p, q) \).

<table>
<thead>
<tr>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Area</td>
<td>1000(km)×1000(km)</td>
</tr>
<tr>
<td>% of service area</td>
<td>30-100</td>
</tr>
<tr>
<td>No. of data objects</td>
<td>10-1000</td>
</tr>
<tr>
<td>Size of data object</td>
<td>256 bytes – 8192 bytes</td>
</tr>
<tr>
<td>Broadcast bandwidth</td>
<td>144kbps</td>
</tr>
<tr>
<td>No. of clients</td>
<td>0-90</td>
</tr>
<tr>
<td>Minimum moving speed of the client</td>
<td>10</td>
</tr>
<tr>
<td>Maximum moving speed of the client</td>
<td>90</td>
</tr>
<tr>
<td>Distance of move</td>
<td>5-50</td>
</tr>
<tr>
<td>Size of ( W_q )</td>
<td>0-5</td>
</tr>
<tr>
<td>Size of ( W_p )</td>
<td>0-5</td>
</tr>
<tr>
<td>No. of broadcast period</td>
<td>50-100</td>
</tr>
<tr>
<td>\text{Size of max. OBC}</td>
<td>Longer than 900m</td>
</tr>
</tbody>
</table>

Table 1. Simulation Parameters
An Efficient Data Dissemination Scheme for Nearest Neighbour Query Processing

The whole geometric service area is divided into groups of MSSs (Mobile Supporting Stations). In this paper, two datasets are used in the experiments (see Figure 3(a)). The first data set $D1$ contains data objects randomly distributed in a square Euclidian space. The second data set $D2$ contains the data objects of hospitals in the Southern California area, which is extracted from the data set in Spatial Datasets (Ref.). Table 1 shows the notations and default parameter settings used in the simulation.

5.1 Latency
In this section, we evaluate the access latencies for various parameter settings such as the client’s speed, the size of the service area, and the numbers of clients. In this paper, we present the Object Boundary Circle (OBC) which represents the distance between the objects as shown in Figure 3(b). The radius of a circle represents a distance between objects and a circle which has the longest radius and is selected as a hot data object such as c and d in Figure 3(b). The server broadcasts data objects with different frequency such as hot and cold data objects (Acharya and Franklin, 1995).

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 is decreased as the size of the service area decreases since the size of the entire broadcast data items is reduced. However, the query arrival time is significantly increased when the client’s speed increases and the client goes outside of the service coverage area, as shown Figure 4(a). In this case, the client’s cached data items become invalid and the client has to tune the broadcast channel again.

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 $D1$. 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 others since most of the client’s speed exceeds the service coverage area as shown in Figure 4(b). In other words, when the time the client spends in crossing a small service area, it is not enough to gather a useful set of local data. Second, we study the performance for different parameters such as

Figure 3: Scope distributions and OBC

(a) Two scope distributions for performance evaluation (b) OBC: (1) $\text{min}_\text{OBC}$: minimum boundary circle is picked as a hot data object. (2) $\text{max}_\text{OBC}$: maximum boundary circle is picked as a hot data object. (3) uniform: all data objects are broadcast in same frequency
An Efficient Data Dissemination Scheme for Nearest Neighbour Query Processing

$min_{OBC}$, $max_{OBC}$ and uniform (see Figure 3(b)) in $D2$. In this experiment, we assume that the clients are uniformly distributed on the map. Figure 4(c) shows the result as the client speed increases from 5 to 50, and Figure 4(d) shows the result as the number of clients increased from 15 to 90.

**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 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 follows:

- **uniform_100%**: The server broadcasts data objects with the same frequency such as flat broadcast in Acharya and Franklin (1995) 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.

Figure 4(e) shows the result as the number of clients is increased from 15 to 90 in $D1$. As shown in the figure, the hot 50% outperforms compared to others as the number of clients increases. Second, we assume that the clients are uniformly distributed in $D2$. Figure 4(f) shows the result as the number of clients increases from 15 to 90. As the figure shows, 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 affects the query response time.
5.2 Cache Hit Ratio
This section evaluates the cache hit ratio for various parameter settings such as the size of $w_p$, the client’s speed and the size of the service area. First, we vary the client’s speed from 10 to 50 in $D_2$. As shown in Figure 5(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 another service area. Second, we vary the client’s speed from 10 to 50 in $D_1$. As shown in Figure 5(b), the number of cache hits decreases as the client’s speed is increases. Third, we vary the value of $w_p$ from 1 to 5 in $D_1$. As shown in Figure 5(c), the number of cache hits increases as the client’s speed decreases and the size of $w_p$ increases.

5.3 Comparison of the Performance of the BBS Scheme and the R-Tree and the D-Tree index
In this section, we compare the BBS scheme with the R-Tree and the D-Tree index. First, we vary the size of the data item from 256 bytes to 8192 bytes in $D_1$ and $D_2$. In this experiment, the server
broadcasts 506 data items periodically to the clients. In $D_2$, we also evaluate BBS with $max_{OBC}$ (see Figure 3(b)). Since the clients do not need to wait and tune to an index segment if they have already identified the nearest object, the BBS shows lower latency compared to the R-Tree and D-Tree index as the data size increases as shown in Figure 6(a). The BBS with $max_{OBC}$ outperforms the R-Tree and D-Tree index and BBS in $D_2$ as shown in Figure 6(b). Second, we vary the number of clients from 50 to 300. As shown in Figure 6(c) and Figure 6(d), the BBS shows lower latency compared to the R-Tree and the D-Tree index in $D_1$ and the BBS with $max_{OBC}$ shows the lowest latency compared to the R-Tree and D-Tree index and BBS in $D_2$.

5.4 Energy Consumption Model
The computing devices in wireless systems usually use batteries as their main energy sources, therefore it is quite important how to use the limited energy efficiently. Let us consider the case of a user who drives a car and wants to find the nearest gas stations with his or her mobile unit, such as a PDA. Here, the user’s PDA has low battery power and, thus, it is necessary to minimize the battery power consumption.

In this paper, there are two energy states such as doze mode, active mode. We now describe the ratio of energy consumption for these states. $E_s$ describes the amount of energy consumption in an energy state $s$ per unit time.

$$E_{DOZE} : E_{ACTIVE} = 1 : ec$$

In this paper, the amount of energy consumed in doze mode for unit time is denoted as unit energy which is 33:16 mW in our experiment. In many processors, the doze mode has extremely low power consumption. In the Hobbit chip from AT&T, for example, the ratio of power consumption in the active mode to the doze mode is 5,000 (Imielinski et al., 1997). In brief, the “ec” stands for energy coefficient which means the active-to-doze ratio, $E_{ACTIVE} / E_{DOZE}$.

In this paper, the average energy consumption can be measured by the amount of unit energy in a given time. In order to choose reasonable coefficients, we should have some reference values. In our experiment, parameter $ec$ is fixed at 48.61. As developing mobile devices are trying to minimize the energy consumption in hardware design, we think these energy coefficients will continuously increase. Table 2 shows the operation of each device according to the energy state.

### Energy Consumption
In this section, we compare the energy consumption for the proposed schemes with $(1,m)$ index. In order to evaluate the energy consumption, both doze time and active time for the client must be estimated. As the result of the following analytic evaluation, the proposed methods significantly reduce the energy consumption compared to $(1,m)$ index method, since it minimizes not only average access time but also average tuning time. Let $AEC_{(1,m)}$ be the average energy consumption of $(1,m)$ index method, and $AEC_{BBS}$ be the average energy consumption of BBS method.

<table>
<thead>
<tr>
<th>Energy state / device</th>
<th>Processor</th>
<th>NIC</th>
<th>GPS receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOZE</td>
<td>doze-off</td>
<td>switch-off</td>
<td>power saving</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>Normal</td>
<td>active</td>
<td>continuous</td>
</tr>
</tbody>
</table>

Table 2: Energy states and corresponding device action
An Efficient Data Dissemination Scheme for Nearest Neighbour Query Processing

Figure 7(a) shows the result as the number of the index $m$ is increased from 1 to 6. As shown in Figure 7(a), if the value of $m$ is greater than or equal to 6, BBS outperforms the $(1,m)$ index. Figure 7(b) and 7(c) show the result as the size of data and number of data items increases respectively. As shown in Figures 7(b) and 7(c), BBS outperforms the $(1,m)$ index, since it reduces not only tuning time but also access latency. That is, the client consumes battery power during both access and tuning time.

6. CONCLUSION

In this paper, we studied the broadcasting and prefetching schemes for LDIS. For broadcasting in LDIS, we presented the BBS and prefetching methods. The BBS method attempts to reduce the access time for the client. Furthermore, the proposed prefetching and OBC can also reduce the query response time and tuning time respectively. We have not changed the previous index schemes, such as R-tree index (Gutman, 1984) and D-tree index (Xu et al., 2004). Rather, we sorted the data objects based on their locations and the server broadcasts the data objects sequentially to the mobile clients. With the proposed schemes, the client can perform the k-NN query processing while it moves without having to tune into the broadcast channel, if the desired data items have already been prefetched into the cache. Therefore, the client can reduce its query response time and the battery power consumption. 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 to the R-tree index and the D-tree since the client does not always have to wait for an index segment.

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

REFERENCES

An Efficient Data Dissemination Scheme for Nearest Neighbour Query Processing


BIOGRAPHICAL NOTES

KwangJin Park received his BSc and MSc degrees in Computer Science from Korea University, Korea in 2000 and 2002, respectively. He is currently a PhD 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.

KwangJin Park