위치 기반 서비스를 위한 데이타 전달 모델
(A Data Dissemination Model for Location-based Services)

박 광 진 † 송 문 배 †† 황 종 선 †††
(Kwangjin Park) (Moonbae Song) (Chong-sun Hwang)

요 약 인덱싱 기법은 무선 환경에서 클라이언트의 선별적인 청취를 지원하기 위하여 사용된다. 즉, 서버는 브로드캐스트 채널을 통해 데이타와 인덱스를 함께 전달함으로써 클라이언트의 선택적인 청취를 지원할 수 있다. 그러나 부가적인 인덱스 정보는 브로드캐스트 주기를 증가시키는 단점이 있다. 위치기반 서비스에서의 질의에 대한 응답지연은 잘못된 결과의 전달이라는 문제점을 가져올 수 있다. 본 논문에서 우리는 무선 브로드캐스트 환경에서 k-인접질의 서비스를 지원하기 위한 BBS(Broadcast Based LDIS Scheme) 기법을 제안한다. BBS 기법에서 서버는 전달하고자 하는 데이타를 위치에 기반을 두어 정렬하며 이를 인덱스 정보와 함께 클라이언트들에게 전달한다. 또한 클라이언트의 질의 처리 수행시간을 단축시키기 위하여 프리패칭과 OBC(Object Boundary Circle)기법을 새로이 제안한다. 제안 논문에 대한 성능평가는 데이타의 분포, 클라이언트의 이동 속도 그리고 서비스 영역의 크기 등 다양한 환경에서 이루어 졌다.

키워드 : 위치기반 서비스, 이동 컴퓨팅, 캐싱, 프리패칭

Abstract Indexing techniques are used to implement selective tuning in wireless environments. Indices are broadcast together with data to help mobile clients locate the required information. As a result, clients stay in doze mode most of the time. The drawback of this solution is that broadcast cycles are lengthened due to additional index information. In location-aware mobile services(LAMSs), it is important to reduce the query response time, since a late query response may contain out-of-date information. In this paper, we present a broadcast-based spatial query processing method (BBS) designed to support k-NN query processing. 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. 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.

Key words : Location Dependent Information Service, Mobile Computing, Caching, Prefetching

1. 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 [1,2]. Using an index can help the client to reduce the amount

† 학생회원 : 고려대학교 컴퓨터과학기술연구소 연구원
kjpark@disys.korea.ac.kr
†† 정 회 원 : 고려대학교 컴퓨터과학기술연구소 연구원
mbsong@disys.korea.ac.kr
††† 종신회원 : 고려대학교 컴퓨터학과 교수
hwang@disys.korea.ac.kr
논문접수 : 2004년 11월 10일
심사완료 : 2005년 3월 21일

of time spent listening to the broadcast channel. 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 10km 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 LAMSs, 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 NN(nearest neighbor) queries in LAMSs. 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 BBS scheme and prefetching method. The performance evaluation is presented in section 4. Finally, section 5 concludes this paper.

2. Background

With the advent of high speed wireless networks and portable device, 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 [1], 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. Air indexing is one of techniques that attempt to address this issue, by interleaving indexing information among the broadcast data items. At the same time, the client can reduce its battery power consumption through the use of select tuning [3,4]. Air indexing 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 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 *Time* is the sum of the *Probe Wait* and Bcast *Wait*. These two factors work against each other [3,4]. There are several indexing techniques such as the distributed indexing approach [3], the signature approach [5], and the hybrid approach [6].

2.2 LDIS Schemes

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 [7], 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 4th, 7th, and 8th only. And they organize each service area as a group in order to reduce the overhead for scope information. In [8], 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. Proposed Algorithms

In this section, we present two schemes for LDIS. We first introduce the broadcast-based LDIS scheme (BBS). In this scheme, the server broadcasts reports which contains 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 data prefetching scheme and OBC in order to reduce the client's tuning time.

3.1 BBS (Broadcast Based LDIS) Scheme

An index gives the ability of selective tuning. The drawback of this solution is that the client has to wait and listen for an index segment in order to identify the nearest data object. In the BBS method, the server broadcast data objects are sorted sequentially according to the location of the data objects. Moreover, based on the distance between the data objects, we assign the difference 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 [9]. For instance, the data objects selected as a hot group will broadcast more frequent than the other groups. We discuss this issue in the section concerning the performance evaluation. 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 nearest object before the associated index segment has arrived. In this method, the structure of the broadcast affects the distribution of the data objects. For example, as shown in Fig. 1, 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:

- *leftmost_P*: a point that is located at the leftmost extremity in the map(e.g., object 'a' in Fig. 1)
- *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'* ≥ *leftmost_P*(e.g., object 'b' in Fig. 1)
- *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
```

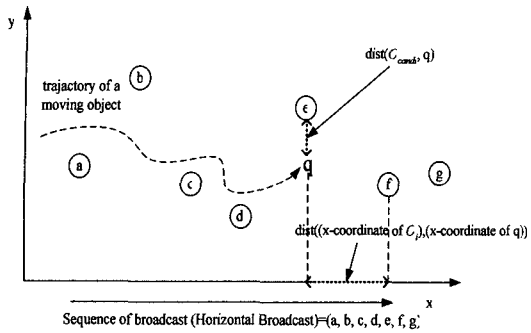


Fig. 1 An example of Horizontal Broadcast

The server decides the sequential order of the broadcast data objects based on the above algorithm. If the final value of *x-dist counter* is bigger than *y-dist counter*, that is, the data objects are horizontally distributed, the server broadcast data objects as a horizontal order. On the other hand, If the value of *y-dist counter* is bigger than *y-dist counter*, that is, the data objects are vertically distributed, the server broadcast data objects as a vertical order.

Notations

- S: server data set
- O: broadcast data object, where $O \in S$
- O_{candi}: candidate for the nearest 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 $O_p \in S$
- q: a query point
- O_{ps}: one of the data items broadcast before O_i in the current broadcast cycle, where $O_{ps} \in S$
- O_p: a data item broadcast just before O_c in the current broadcast cycle, where $O_p \in S$
- O_f : the client's first tuned data item in the broadcast channel
- Data first: the server's first broadcast data item in the current broadcast cycle
- Flag A: if the dist(x-coordinate of O_{ps}, x-coordinate of q) is larger than dist(O_c, q), then set to 1 (initially set to 0). This flag guarantees that the client dose not miss the NN in the current broadcast channel
- Flag B: if the dist((x-coordinate of O_c), (x- coordinate of q)) < dist(O_{candi}, q)), then set to 1 (initially set to 0),(see lemma 1)

Lemma 1: While the data objects are sequentially broadcasted 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-coordinate of O_i), (x-coordinate of q))>dist(O_{candi}, q), 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' and O_c is an object 'f', as shown in the Fig 1. If dist((x-coordinate of the object 'f'), (x-coordinate of 'q'))>dist('e','q'), then the objects 'f' and 'g' are located outside of the NN range and thus the client stop tuning the broadcast channel and select the object 'e' as the NN. There are two cases in which the clients tune to the broadcast channel.

Case 1: the client could not identify the NN in the current broadcast cycle, since it was not able to determine whether or not the desired data item was broadcasted before it tuned to the broadcast channel(see Fig. 2).

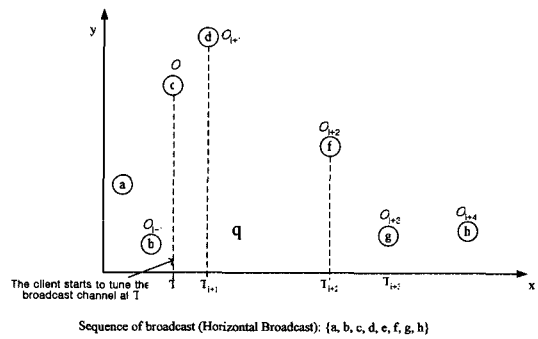


Fig. 2 the client could not identify the NN in the current broadcast cycle

Lemma 2: While Flag A=0 and $O_f \neq$ Data first, the client could not identify the NN in the current broadcast cycle.

Proof: Let the client begin to tune at time T_i. At T_i, $O_c=O_i$ and O_{candi} is O_i(see Fig. 2. At T_{i+1}, $O_c=O_{i+1}$ and O_{candi} is O_i, since dist(O_i, q)<dist(O_{i+1}, q). At T_{i+2}, $O_c=O_{i+2}$ and O_{candi} is O_{i+2}, since dist (O_{i+2}, q)< dist(O_i, q) and dist(O_{i+2}, q)< dist(O_{i+1}, q). At T_{i+3}, $O_c=O_{i+3}$ and O_{candi} is O_{i+2}, since dist(O_{i+2}, q)< dist(O_{i+3}, q). Then, the client stops tuning to

the broadcast channel, since dist(O_{i+2}, q)<dist(x-coordinate of O_{i+3}, x-coordinate of q)(see lemma 1). However, the client could not guarantee O_{i+2} as the NN, since it missed the O_{i-n}, such as O_{i-1} data item, and one of them could also be O_{candi}. Thus, if Flag A=0 and Flag B=1, then the client stops tuning to the broadcast channel and switches to doze mode, until the next index segment arrives.

Case 2: the client is able to identify the NN from the current broadcast cycle, since it is sure that the desired data item is going to appear in the current broadcast cycle(see Fig. 3).

Lemma 3: If Flag A is set to 1, then the client can identify the NN from the current broadcast cycle.

Proof: Let O_c be data item 'c' in Fig. 3. After the client receives data item 'd', Flag A is set to 1, which means that data item 'c' and the data items before 'c', such as 'a' and 'b', are not candidates for the NN, since the distances from the x-coordinates of 'a', 'b, and 'c' to the x-coordinate of q are longer than dist(d, q). Consequently, we can conclude that the client does not miss the NN and can find it from the current broadcast channel.

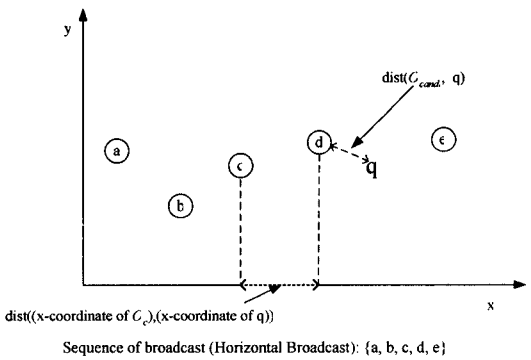


Fig. 3 The client is able to identify the NN in the current broadcast cycle

Definition 1: If O_c is Data first, then the client can identify the NN from the current broadcast cycle.

We assume that each client has a queue, in order to maintain previously broadcasted data objects. The client uses the following algorithm to identity the NN.

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(optimal tuning is required)
2:   then read the first broadcast data item and go
     to doze mode until the index segment arrives
3:   return NN using index segment
4: else if (optimal latency is required) {
5:   do {
6:     for each data object O ∈ S
7:       if(Flag A=0 and Flag B=1)
8:           then stop tune and go to doze until the
             next index segment arrives
9:       else if (Oc is the first broadcast data object)
10:        then Oc = Ocandi
11:      else if dist(Oc, q)< dist(Op, q)
12:        then Oc = Ocandi
13:      else Op = Ocandi
14:  } while (getting to the index segment or Flag
     B=1)
15: Ocandi=NN
16: return NN
```

3.2 Prefetching Scheme

The result of the *NN* query is changed if the client moves. Thus, the client has to tune its broadcast channel every time it moves. Data prefetching has been proposed as a technique for hiding the access latency of data item that defeat caching strategies. 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. And we denote the Euclidian distance between the two points p and q by $dist(p, q)$. In the map, we have $dist(p, q) := \sqrt{(px-px)^2 + (py-py)^2}$

Let $\boldsymbol{P}$:= { p_{-n} ..., p_{-2}, p_{-1}, p_0, p_1, p_2 ..., p_n} be a set of n distinct points that represent the data

objects, and q represents a query point.

Notations:

w, $n \geq 0$ and $(w-n) \geq 0$

target= an object p_0, where $p_0 \neq p_n$ and $\{p_{-n}, p_0, p_n\} \in P$ then $dist(p_0, q) \leq dist(\forall p_{-n}, q)$ or $dist(p_0, q) \leq dist(\forall p_n, q)$

p_{min} = an object p_{-w}, where $dist(p_{-(w-n)}, q) \leq dist(p_{-w}, q) \leq dist(p_{-(w+n)}, q)$

p_{max} = an object p_w, where $dist(p_{w-n}, q) \leq dist(p_w, q) \leq dist(p_{w+n}, q)$

A query can be categorized as the nearest or the k-nearest based on the number of returned objects. 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 $2n + 1$. Hence, w_p= set of $2n + 1$ points. For instance, 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). 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= p0 and
       prefetch a data object from pmin to pmax }
5:   else
6:     wait until the desired data comes from the
       server
7: }
8: doze mode
```

Lemma 4: 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 $pi \in S_q$.

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

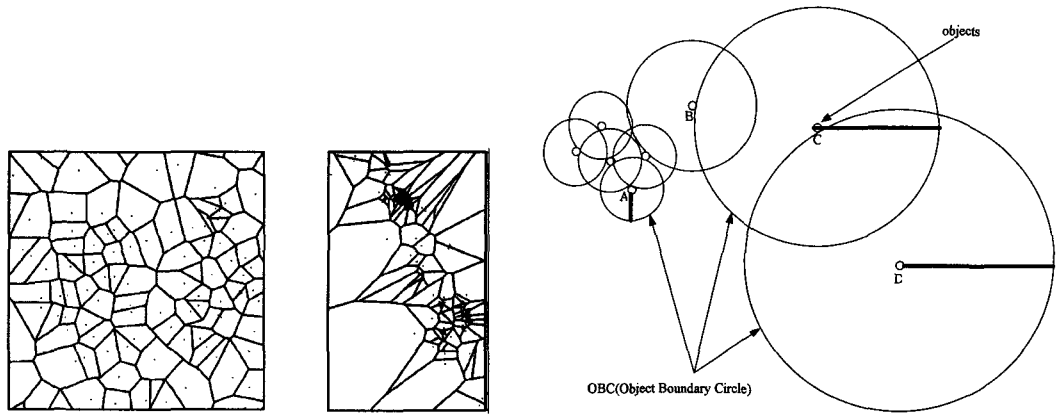
4. Performance Evaluations

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_p and the service coverage area. Finally, we compare the performance of the BBS scheme and the R-Tree index[12] scheme and the D-tree index[11] 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 [8,12]. The distance can be computed using the Euclidian distance between the two points p and q as defined by $dist(p, q)$. 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 Fig. 4(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 at [13]. Table 1

Table 1 Simulation Parameters

Description	Setting
Service area	1000(km)*1000(km)
% of service area	30-100
No. of data items	10-100
Size of data objects	256bytes -8192bytes
Broadcast bandwidth	144bps
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 Wq	0-5
Size of Wp	0-5
No. of broadcast period	50-100
Size of max_OBC	Longer than 900m



(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

Fig. 4 Scope distributions and OBC

shows the notations and default parameter settings used in the simulation.

4.1 Latency

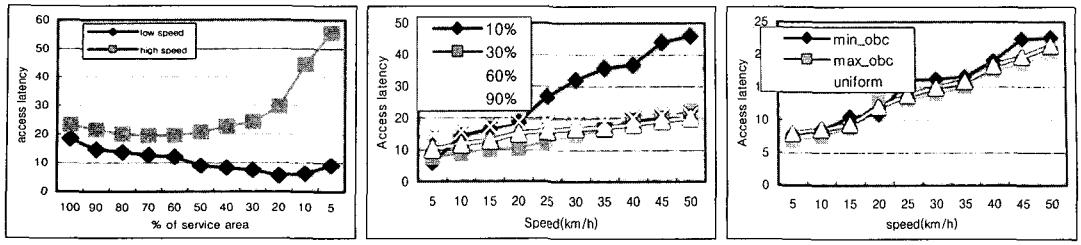
In this section, we evaluate the access latencies for various parameter settings such as the client's speed, the size of the service area, and the numbers of clients. In this paper, we present the Object Boundary Circle (OBC) which represents the distance between the objects as shown in Fig. 4(b). The radius of circle represents a distance between objects. And a circle which has the longest radius is selected as a hot data object such as *c* and *d* in Fig. 4(b). The server broadcasts data objects with different frequency such as hot and cold data objects [9].

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 increase and the client goes outside of the service coverage area, as shown Fig. 5(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 *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 others since the most of the client's speed exceeds the service coverage area as shown in Fig. 5(b). Second, we study the performance for different parameters such as *min OBC*, *max OBC* and uniform (see Fig. 4(b)) in *D*2. In this experiment, we assume that the clients are uniformly distributed in the map. Fig. 5(c) shows the result as the client speed increases from 5 to 50.

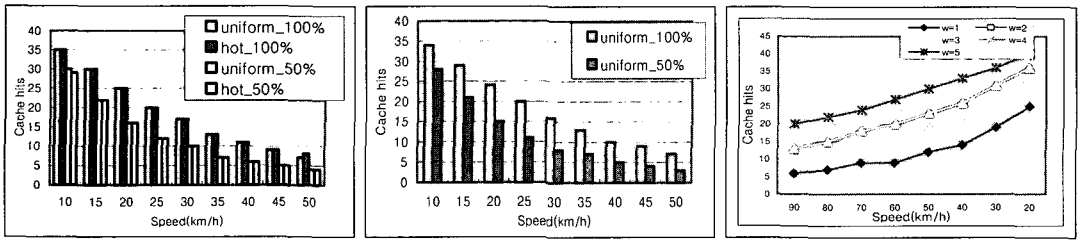
Effect of the distribution of data objects and the clients' location In this section, we study the effect of the distributions of the data objects and the clients' location. First, we assume that the clients are crowded in a specific region such as downtown. Those data objects which are located in such 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 at broadcast in [9] and the service coverage area is



(a) (b) (c)

Fig. 5 Access latency



(a) (b) (c)

Fig. 6 Cache hit ratio

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.

4.2 Cache Hit Ratio

This section evaluates the cache hit ratio for various parameters 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 *D2*. As shown in Fig. 6(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 *D1*. As shown in Fig. 6(b), *the number of cache hits decreases as the client's* speed is increased. Third, we vary the value of *wp* from 1 to 5 in *D1*. As shown in Fig. 6(c), the number of cache hits increases as the client's speed is decreased and the size of *wp* increases.

4.3 Comparison of the Performance of the BBS Scheme with 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 *D1* and *D2*. In this experiment, the server broadcast 506 data items periodically to the clients. In *D2*, we also evaluate BBS with *max_OBC* (see Fig. 4(b)). Since the clients do not need to wait and tune an index segment if they have already identified the nearest object, the BBS shows lower latency compare to the R-Tree and the D-Tree index as the data size increases as shown in Fig. 7(a). The BBS with *max_OBC* outperform the R-Tree and the D-Tree index and BBS in *D2* as shown in Fig. 7(b). Second, we vary the number of clients from 50 to 300. As shown in Fig. 7(c) and Fig. 7(d), the BBS shows lower latency compare to the R-Tree and the D-Tree

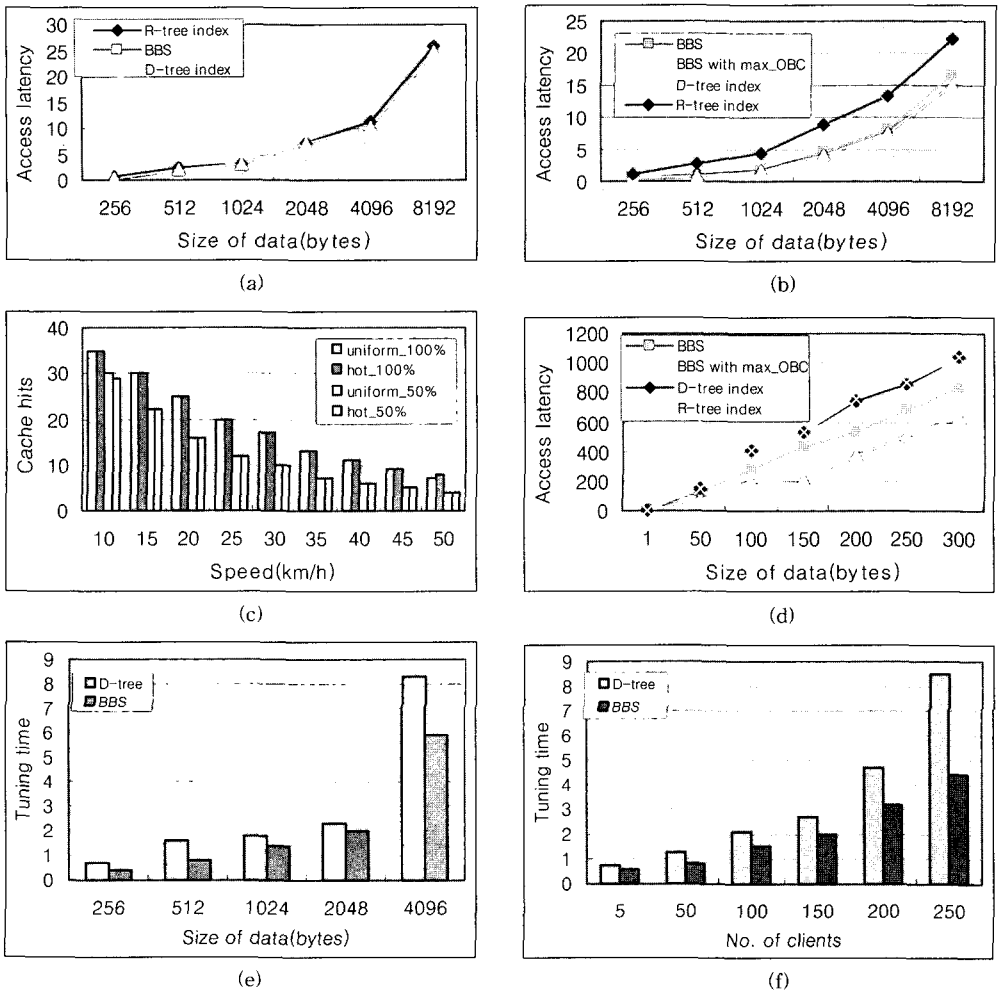
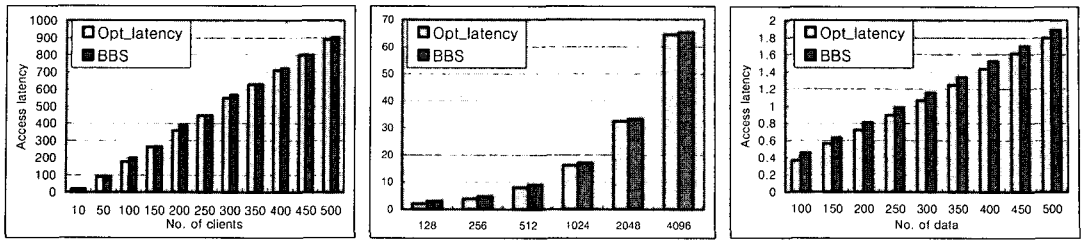


Fig. 7 Compare the Performance of BBS Scheme with the R-Tree and the D-Tree Index

index in *D1* and the BBS with *max_OBC* shows the lowest latency compared to the R-Tree and the D-Tree index and BBS in *D2*. Third, we compare the tuning time of the BBS with the D-tree index. In this experiment, we assume that the index segment is broadcasted once at the beginning of the broadcast cycle. Fig. 7(e) shows the result of the tuning time between the BBS and the D-tree index as the sizes of the data items are increased from 256bytes to 4096bytes. As shown in the figure, the BBS spent lower tuning time than the D-Tree index, since the client do not need to wait and tune anymore if they have already identified the nearest object, while the D-Tree index scheme has to wait an index segment even if the desire data item is come to the broadcast channel. Finally we compare the tuning time of the BBS and the D-Tree index as the number of client increases from 5 to 250. Fig. 7(f) shows the result.

Comparison of BBS with optimal_latency In this section, we study the access latency with the BBS and opt_latency(optimal access latency) in *D1*. In this experiment, we assume that the clients are uniformly distributed in the map. Fig. 8(a) and Fig. 8(b) show the results of the access times as the number of clients and the size of data increase respectively. Fig. 8(c) shows the result of the access time between the opt_tune and BBS as the



(a) (b) (c)

Fig. 8 Comparison of BBS with optimal latency

number of data increase. As shown in the figure, BBS performs closely as same as the opt_latency.

5. Conclusions

In this paper, we studied the broadcasting and prefetching schemes for LDIS. For broadcasting in LDIS, we present the BBS and prefetching methods. The BBS method attempts to reduce the access and the tuning time for the client. Furthermore, the proposed prefetching and OBC can also reduce the query response time and the tuning time respectively. We do not change the previous index schemes, such as R-tree index [10] and D-tree index [2,11]. Rather, we sort 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 performs the k-*NN* query processing while it moves without having to tune 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 and the tuning time 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.

참 고 문 헌

[1] Dik Lun Lee, Jianliang Xu, and Baihua Zheng, "Data Management in Location-Dependent Information Services," *IEEE Pervasive Computing*, 1(3), 2002.

[2] J. Xu, B. Zheng, W.-C. Lee, and D. L. Lee, "Energy E±cient Index for Querying Location-Dependent Data in Mobile Broadcast Environments," In *Proc. of ICDE*, 2003.

[3] T. Imielinski, S. Viswanathan, and B.R.Badrinath, "Data on Air: Organization and Access," *IEEE Trans. Knowledge and Data Eng*, 9(3), 1997.

[4] T. Imielinski, S. Viswanathan, and B.R.Badrinath, "Energy efficient indexing on air," In *Proc. Of SIGMOD*, 1994.

[5] W. C. Lee and D. L. Lee, "Using signature techniques for information filtering in wireless and mobile environments," *Distributed and Parallel Databases*, 4(3), 1996.

[6] Q. L. Hu, W.-C. Lee, and D. L. Lee, "A hybrid index technique for power efficient data broadcast," *Distributed and Parallel Databases*, 9(2), 2001.

[7] Jianliang Xu, Xueyan Tang, and Dik Lun Lee, "Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments," *IEEE Trans. Knowledge and Data Eng*, 15(2), 2003.

[8] Baihua Zheng, Jianliang Xu, and Dik L. Lee, "Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments," *IEEE Trans. Comp.*, 51(10), 2002.

[9] S. Acharya and Michael Franklin, "Broadcast Disks: Data Management for asymmetric communication environments," In *Proc. of SIGMOD*, 1995.

[10] A. Guttman, "R-trees: A dynamic index structure for spatial searching," In *Proc. of SIGMOD*, 1984.

[11] 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," In *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 2004.

[12] Daniel Barbara, "Sleepers and Workaholics: Cashing Strategies in Mobile Environments," In *Proc. Of SIGMOD*, 1994.

[13] Spatial Datasets, http://dias.cti.gr/~ytheod/research/datasets/spatial.html.

[14] N. Roussopoulos, S. Kelley, and F. Vincent. "Nearest neighbor queries," In *Proc. of SIGMOD*, 1995.

[15] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Location-based Spatial Queries," In *Proc. Of SIGMOD*, 2003.

박 광 진
정보과학회논문지 : 데이타베이스
제 32 권 제 3 호 참조

송 문 배
정보과학회논문지 : 데이타베이스
제 32 권 제 3 호 참조

황 종 선
정보과학회논문지 : 데이타베이스
제 32 권 제 3 호 참조