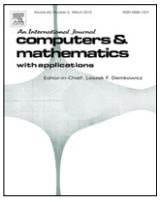
Computers and Mathematics with Applications 64 (2012) 1242–1251

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa



MLAIN: Multi-leveled air indexing scheme in non-flat wireless data broadcast for efficient window query processing

Seokjin Im, JinTak Choi *

College of Information and Technology, University of Incheon, Incheon, Republic of Korea

ARTICLE INFO

Keywords:
Non-flat wireless data broadcast
Distributed indexing scheme
Spatial data
Window query

ABSTRACT

In ubiquitous computing, it is critical to allow a great number of clients to access information simultaneously at any place and at any time. Wireless data broadcasting provides effective information services due to its own high scalability. In this paper, we propose a novel indexing scheme for spatial data items that adopts multi-leveled grid partition to support window queries in non-flat data broadcasting that considers clients' skewed data access patterns. In the proposed scheme, each cell of the partition is restricted to its number of data items. This constraint makes the proposed scheme different from other schemes that use a space partition. Cell indexes of cells that keep link information between cells are interleaved with data items on the channel. The scheme allows clients to access queried data items quickly by reducing the broadcast cycle and the spacing between indexes, and providing multiple paths to a cell on the channel. We show the efficiency of the proposed scheme with regard to the access time, tuning time, and energy consumption using intensive simulation studies.

1. Introduction

Ubiquitous computing has evolved tremendously with technological advances in wireless networks and context aware systems and is becoming more and more pervasive in daily life [1–4]. In the environment, numerous mobile clients try to access a variety of information at any place and at any time while moving freely. It is critical to provide clients with information services efficiently regardless of the number of clients [5–11].

A wireless data broadcasting system that enables a server to disseminate data items periodically over a wireless channel provides effective information services, in addition to offering high scalability because it can accommodate an arbitrary number of mobile clients simultaneously [5]. In such a system, each client accesses the channel and downloads the desired data items. Microsoft's MSN Direct Services, which provide information to clients, runs using wireless data broadcasting [12].

In particular, the system broadcasting spatial data items, which have geographic location information regarding the features of a particular area, such as data on hotels in a city, provides numerous clients with efficient location dependent information services. Using the system, each client independently processes its own spatial query, such as a window query to find data items within a given query window or a kNN query to find k nearest neighbors from a given query point, by monitoring the wireless channel. For example, clients can find hotels within a 100 m square centered at their current location in a metropolitan area such as New York City.

Due to the limited battery life of clients, they operate in two modes: active mode (the energy-consuming mode that listens to the channel to download data items) and doze mode (the energy-saving mode). In order to download data items satisfying

* Corresponding author.
E-mail address: choi@incheon.ac.kr (J. Choi).

0898-1221/$ – see front matter
doi:10.1016/j.camwa.2012.03.068

a given query from the wireless channel, clients must listen to the channel until the items appear on the channel. This causes clients to consume a great deal of energy during data retrieval because of the long stay in active mode that is required.

Air indexing schemes have been proposed for energy efficient query processing by letting clients selectively listen to only the data items queried. An air index for spatial data items stores the location information about the items and their broadcasting times. The index is then interleaved with the items on the channel. After accessing the index, clients filter out the queried items and selectively download them with their broadcasting times.

The performance of the system is characterized by access time and tuning time [5]. Access time refers to the time elapsed from the beginning of processing a given query to the receipt of the answer to the query. Tuning time is the amount time necessary in active mode for the access time.

According to the uniformity of the broadcast frequency of data items, the wireless data broadcasting systems can be categorized into flat data and non-flat data. In flat data broadcasting, the server disseminates all data items with the same frequency in a broadcast cycle [5]. On the other hand, in non-flat data broadcasting, the server disseminates popular data items, called hot data items, more frequently than regular ones in a broadcast cycle [13]. When clients' data access patterns are skewed to hot data items, non-flat data broadcasting reduces the average access time because clients can receive hot data items more quickly than regular ones on the channel.

Various data scheduling and air indexing schemes for spatial data have been developed for efficient spatial query processing. In flat data broadcasting, to support spatial queries, a Hilbert curve index (HCI) and distributed spatial index (DSI) were proposed on a Hilbert curve (HC) in [14,6], respectively. The two indexing schemes cause clients to spend an excessive amount of time accessing queried data items; these two indexing schemes extract data items by listening to many candidates that were decided with HC values for each item rather than their real coordinates. In [7], a cell-based distributed index (CEDI) using a regular cell partition was proposed for time efficient query processing. With CEDI, clients listen to only their desired data items by extracting them using their real coordinates. Access time is longer in CEDI when data items are skewed in several cells because the difference between the distributed indexes increases. The three works on spatial data can cause the deterioration of client performance in skewed data access patterns to hot items because of flat data broadcasting. In non-flat data broadcasting, a grid-based distributed index for non-flat broadcast (GDIN) organized over a regular space partition was proposed to reduce the access time when clients' data access pattern's skewed to hot data [8]. GDIN repeats hot cells, which are cells containing one or more hot data items as well as regular ones, several times in a broadcast cycle. GDIN rapidly increases the number of data items repeated in a cycle when numerous regular data items are skewed around hot data items in a hot cell because GDIN repeats all regular items in a hot cell at the same frequently as with the hot items. This lengthens GDIN's access time because the length of a broadcast cycle gets longer and the differences between indexes on the broadcast channel also increases.

In this paper, we focus on non-flat data broadcasting with spatial data items, proposing a multi-leveled air indexing scheme in non-flat data broadcasting (MLAIN) to efficiently process window queries with the popularity of spatial data items considered. MLAIN partitions the data space by recursively subdividing it into four quadrant cells of multiple levels until all cells satisfy the constraint that the number of data items in a cell is not over a specified number. It is this constraint that makes MLAIN different from other indexing schemes using a regular space partition. Each cell having one or more hot data items, called a hot cell, is broadcast more frequently than regular cells to help clients quickly access hot data items on the channel. A cell index for each cell satisfying the constraint is interleaved with data items in the cell on the wireless channel in the distributed manner. Using the indexes on the channel, clients can process their own queries and download desired items from the channel.

MLAIN contributes to improved access time in the following ways:

- Reducing the number of regular items repeated with hot data items through the use of the constraint. This shortens the length of a broadcast cycle. Therefore, clients can quickly access both hot data items and regular ones because the average access time is half the cycle under the assumption that the probability distribution of the access to the channel of clients is uniform within a cycle.
- There is shortened spacing between indexes because of the constraint, although lots of regular data items are skewed around hot items, unlike a regular space partition. This enables clients to access index information quickly and to start processing a given query immediately.
- There are multiple access paths to a cell on the channel. A MLAIN cell index keeps link information that clients can access in the upper level, same level, and lower level. The link information provides clients with multiple paths to a cell. The paths enable clients to improve the access time because they can take another path to access a cell although they miss a path to the cell, rather than waiting for the next cycle.

For the tuning time, MLAIN allows clients to not spend an excessive amount of time in active mode accessing data items because it lets them extract their desired data items using the real coordinates of spatial data items before downloading data items from the wireless channel. In addition, MLAIN employs a linear table structure to match clients' linear channel access patterns.

The rest of the paper is organized as follows: Section 2 reviews the related works in wireless data broadcasting. We present the proposed MLAIN in Section 3. Then, we describe simulation environments and evaluate the performance of the proposed scheme in Section 4. Finally, we conclude the paper in Section 5.

2. Related works

In mobile computing environments, information can generally be delivered in two methods: on-demand and broadcasting. With the on-demand method, a client requests data items from a server and the server sends the data items to the client. Broadcasting disseminates data items periodically over a wireless channel and clients download the data items they want from the channel. Wireless data broadcasting has been widely researched due to its high scalability supporting a great of number of clients simultaneously. Here, we briefly discuss the works related to data broadcasting.

2.1. Air indexing

Air indexing schemes are adopted to reduce the tuning time by selectively listening to the data items the clients desire. In these schemes, an index with the broadcasting time of data items is interleaved with data items on the wireless channel. Clients can then predict when desired items appear on the channel after accessing the index. It means that clients can stay in doze mode for energy-saving until the desired items appear on the channel. Clients process a given query in two phases:

- Initial probe: After tuning in the wireless channel, a client determines the time when the index appears. Then, it switches to the doze mode, waiting for the index. Probe wait means the duration from tuning in to accessing to the index.
- Index search and data download: The client awakes to the active mode when the index appears on the channel. Then, it filters out the queried data items with the index and determines the time when the data items will appear on the channel. The client switches to doze mode until the data items appear; it then awakes and downloads the data items. Bcast wait means the duration from the access to the index to downloading all queried data items.

The access time is the sum of probe wait and bcast wait. The tuning time is the amount of time staying in active mode for the access time.

2.2. Flat and non-flat data broadcasting

For efficient location dependent information services in wireless data broadcasting, a variety of researches have studied on air indexing schemes in flat and non-flat data broadcasting. Here, we briefly discuss the schemes to process given spatial queries efficiently.

In the flat data broadcasting of spatial data items, various indexes have been proposed for window queries and kNN queries. In [14], the authors propose the HCI to efficiently support window queries as well as kNN queries that employs B+ tree constructed on HC. The HCI uses HC values to represent the locations of data items. This causes clients to consume excessive energy during query processing because they have to access numerous candidate items decided by HC values, not the original coordinates, before extracting the queried items. The DSI over a HC was proposed that adopts a distributed table to allow for a quick start of query processing [6]. It also supports two kinds of queries, i.e., window queries and kNN queries. It shows improved performances compared to the HCI. However, it still causes clients to consume a large amount of energy for the same reasons as the HCI.

While the HCI and DSI use HC, indexes based on space partition and R-tree were proposed for window queries and kNN queries. In [7], the authors proposed CEDI with two-level tables indexing the regular grid space for window queries. It enables clients to process given queries energy efficiently by letting them listen to only the queried data items using their original coordinates. The schemes mentioned above cause long access times when clients' data access patterns are skewed because they are based on flat broadcasting. The authors proposed a cell-based hybrid index (CHI) on the regular grid partition to support kNN queries [10]. The CHI provides global knowledge on data distribution to quickly decide the search space guaranteeing kNN and local knowledge for efficient pruning of data items from the space. In [11], a bR-tree was proposed as a variant index structure of the R-tree to support kNN queries efficiently using the bit-vector information of data items. Further, in [15], the authors proposed spatial query containment for processing window queries and kNN queries semantically. This reduces computing loads due to utilizing the results client have.

To consider clients' nonuniform data access distribution, non-flat data broadcasting is widely adopted. It enables clients to access hot data items more quickly than regular data items. In [13], broadcast disk scheduling data on a wireless channel was proposed to consider clients' skewed data access distribution. According to the access rate, data items are grouped into several logical disks that are assigned to different broadcast frequencies. Popular items' disks have higher broadcast frequencies than the others and data items are broadcast by circularly taking the items from the disks according to their relative frequencies. This allows clients to quickly access popular items on the channel. The authors proposed an optimal broadcast scheduling approach with the nonuniform data access pattern of the clients considered in [16]. Time-critical scheduling of data items was also proposed in [17]. Recently, in [18], the authors proposed MHash, an air indexing scheme in non-flat broadcasting aimed at the access time optimized for energy-efficient query processing. These works mentioned focus on non-spatial data in non-flat broadcasting.

For the non-flat data broadcasting of spatial data items, GDIN was developed in non-flat broadcasting using a regular space partition. The scheme shows much shorter access time than the other flat broadcast schemes when clients access hot data items more frequently. GDIN, however, causes the access time to be lengthened when numerous regular data items are around hot items because the regular data items in a hot cell containing hot items are broadcast with the same frequency as

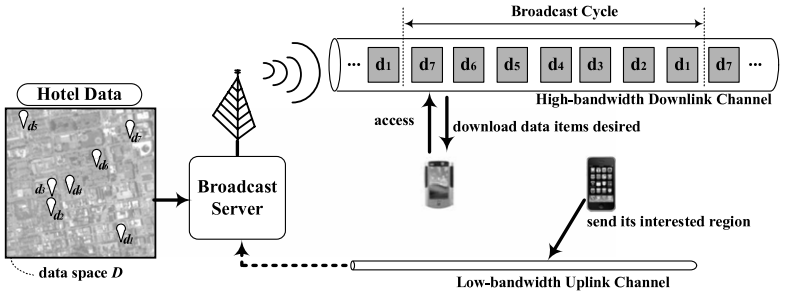
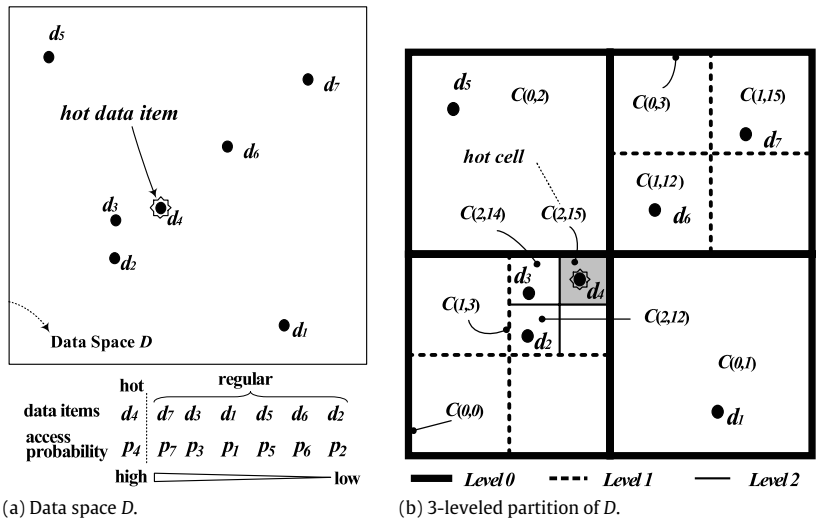


Fig. 1. The broadcast system for the proposed MLAIN.



(a) Data space *D*. (b) 3-leveled partition of *D*.

Fig. 2. Data space *D* and its grid partition.

the hot items. Reducing the number of regular data items in a hot cell improves the access time. This motivates us to propose MLAIN based on a multi-leveled space partition, rather than a regular space partition. In this paper, we develop MLAIN for processing window queries efficiently in the non-flat broadcasting of spatial data.

3. Proposed indexing scheme

As the environment for the proposed MLAIN, we consider a broadcast system disseminating N spatial data items, $\{d_1, \ldots, d_N\}$, on a two-dimensional region, called data space D, with a high-bandwidth downlink channel for broadcasting and a low-bandwidth uplink channel for updating the access probability of each data item with the clients' interested regions. Here, we set D to a two-dimensional region because we restrict spatial data items as ones whose locations are described with the longitude and latitude on the ground. Fig. 1, for example, depicts the system broadcasting data on seven hotels in a city via the downlink channel. Clients access the channel and then download the desired data items. They also send their interested regions to the server via the uplink channel in order for the access probability of data items to be computed.

Using each client's interested region R_{int} sent to the server, the access probability p_i of data item d_i of N data items on D, is computed as $f_i/(\sum_{j=1}^{N} f_j)$, where f_i is the number of R_{int} of all clients overlapped with d_i. With N data sorting decreasingly in p_i, we define hot data as the first N_h data items, where $N_h = \lfloor N \cdot H_r \rfloor$ and H_r is a ratio for deciding the number of hot data items. For example, Fig. 2(a) shows seven items $\{d_1, \ldots, d_7\}$ on D depicted in Fig. 1 and their access probability. With $H_r = 0.15$, d_4 with the highest probability is selected as a hot data item.

To apply MLAIN, we partition D by recursively subdividing it into four quadrants until the number of data items in each quadrant is equal or less than the maximum capacity, m_c, the maximum number of data items each quadrant can contain. For example, Fig. 2(b) shows a three-leveled partition with $m_c = 1$, where level 0 is the highest. We denote a grid cell as $C(l, k)$, where l is the level depth and k the grid cell number on the level, $0 \leq k < 2^{2(l+1)}$, following Z-order. We also define a hot cell as a grid cell with hot data items, which are satisfied with the constraint with m_c, such as $C(2, 15)$ in Fig. 2(b). Using the multi-leveled partition with m_c constraint, MLAIN avoids the shortcomings of the regular partition mentioned above.

3.1. Index structure

MLAIN has two kinds of indexes: a cell index *CI* for each cell with data items on each level and a hot cell index *HI* for hot cells.

CI for $C(l, k)$ is defined as a table that keeps three kinds of information about: what cells are a sibling and child of $C(l, k)$, the broadcasting time of the cells, and the data items in $C(l, k)$. $CI(l, k)$ for $C(l, k)$ is organized as follows:

$$CI(l, k) = \langle t_h, t_r, ST, CT, DT \rangle \tag{1}$$

- t_h: the broadcasting time of the hot cell index encountered next on the channel.
- t_r: the broadcasting time of a grid cell on the top level encountered next on the channel.
- *ST* (sibling table): $\{\langle k_s, t\rangle | \lfloor k/4 \rfloor * 4 \leq k_s < (\lfloor k/4 \rfloor + 1) * 4$ and t is the time when $C(l, k_s)$ is broadcast on the channel$\}$, where t is null when $C(l, k_s)$ has no data.
- *CT* (child table): $\{\langle k_c, t\rangle | k * 4 \leq k_c < (k + 1) * 4$ and t is the time when $C(l + 1, k_c)$ is broadcast on the channel$\}$, where t is null when $C(l + 1, k_c)$ has no data.
- *DT* (data table): $\{\langle (d_x, d_y), t\rangle | (d_x, d_y)$ means the original coordinate of data item, d in $C(l, k)$, and t is the time when d is broadcast on the channel$\}$. *DT* is null when the number of items in $C(l, k)$ is greater than m_c.

t_h helps clients access the hot cell index directly without referring to other indexes. t_r enables clients to access a cell on the top level from any other level. *ST* and *CT* keep the information on the sibling and child cells with data items, respectively, and give the clients an effective way to access the desired sibling cells or child cells of $C(l, k)$ directly. *DT* makes clients extract the items of $C(l, k)$ contained within a given query window before accessing them.

HI is defined as a table that keeps information about what cells are hot cells and the broadcasting time of the hot cells. *HI* is organized as follows:

$$HI = \langle t_r, HT \rangle. \tag{2}$$

- *HT* (hot cell table): $\{\langle (l, k), t\rangle | (l, k)$ means hot cell $C(l, k)$ and t is the time when $CI(l, k)$ is broadcast on the channel$\}$.

HT gives clients information regarding which cells are hot cells and when the cells appear on the channel. With *HT*, clients access hot cells successively on the channel.

3.2. Channel structure

With the proposed MLAIN, each cell is organized with its own cell index and data items within it and is placed on the channel according to l and k, top level to bottom, and low cell number to high in a level. Each cell is also followed by its child cells.

HI and all hot cells are repeated in front of every $\lfloor (N - N_h)/freq \rfloor$ data items on the channel, where *freq* is the frequency repeating hot cells.

For example, Fig. 3 shows the structure of the downlink channel applying MLAIN with $freq = 3$ and $CI(0, 0)$ and *HI*. As shown, $C(0, 0)$ is followed by its own child cells $C(1, 3)$, $C(2, 12)$, $C(2, 14)$, and $C(2, 15)$. t_h of $CI(0, 0)$ keeps t_2, the time when the hot cell index appears on the channel right after $CI(0, 0)$. t_r of $CI(0, 0)$ keeps t_3, the time when the grid cell of the top level appears on the channel right after $CI(0, 0)$. *ST* of $CI(0, 0)$ holds the information regarding the sibling cells of $C(0, 0)$ and their broadcasting times on the channel. *CT* keeps the information on $C(1, 3)$, the child cell of $C(0, 0)$, and its broadcasting time t_1. Here, *DT* is null because $C(0, 0)$ has more data items than m_c.

HI shown in Fig. 3 holds t_r and *HT*, which keeps the information on hot cell $C(2, 15)$ and its broadcasting time t_5.

3.3. Window query processing

With MLAIN, each client processes its own window query with query window qw by following three steps. qw is a rectangle defined by (L_x, L_y) and (U_x, U_y), which are the coordinates of the lower-left and upper-right corner, respectively.

- *Step* 1. Each client determines Q, which is a set of cells on the bottom level l' overlapped with qw, as follows:

$$Q = \{C(l', k') | k' = cellNumber\ (X, Y)\} \tag{3}$$

for $\lfloor \frac{L_x}{\delta} \rfloor \leq X \leq \lfloor \frac{U_x}{\delta} \rfloor$ and $\lfloor \frac{L_y}{\delta} \rfloor \leq Y \leq \lfloor \frac{U_y}{\delta} \rfloor$.

Here, $cellNumber\ (X, Y) = \Sigma_{l=0}^{l'} (2Y_l + X_l) 2^{2(l'-l)}$, $Y_l = \lfloor \frac{(Y \% 2^{l'-l+1})}{2^{l'-l}} \rfloor$ and $X_l = \lfloor \frac{(X \% 2^{l'-l+1})}{2^{l'-l}} \rfloor$, and δ is the length of a side of a cell on the bottom level l'.

- *Step* 2. In this step, the client accesses hot cells including the cells of Q in order to download the data items of the hot cells belonging to qw. The step begins with accessing *HI* on the channel. To check that cell $C(l, k)$ is overlapped with qw, we define the set of children of $C(l, k)$ on level l' as follows: $S_{ch}(l, k)$, as $\{C(l', k') |, k2^{2(l'-l)} \leq k' < (k + 1)2^{2(l'-l)}\}$.
  ◇ **getTimeWithHT**.

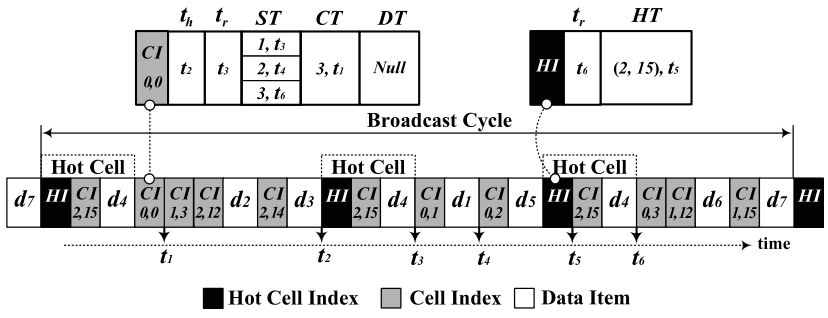


Fig. 3. The structure of the wireless channel applying MLAIN.

Using *HT* of *HI* accessed, the client receives the broadcasting time of all hot cells overlapped with qw.

For each entry $\langle CI(l, k), t\rangle$ in *HT* of *HI*,

$$\{\text{if } (S_{ch}(l, k) \cap Q) \neq \phi \text{ then } ctQueue \leftarrow t\}$$

where *ctQueue* is a queue keeping the broadcasting time $t(\in HT)$for cells in increasing order.

◇ **filterDataWithDT**.

Accessing each $CI(l, k)$ at each time dequeued from *ctQueue*, the client filters out data items belonging to qw with *DT* of $CI(l, k)$.

For each $CI(l, k)$ accessed,

$$\begin{cases} \text{if } (S_{ch}(l, k) \cap Q) \neq \phi \quad \text{and} \quad DT \neq \text{NULL} \\ \text{for all } \langle (d_x, d_y), \quad t\rangle (\in DT) \\ \text{if } (d_x, d_y) \in qw \quad \text{then } dtQueue \leftarrow t(\in DT) \\ \text{subtract } S_{ch}(l, k) \quad \text{from } Q. \end{cases}$$

Here, *dtQueue* is a queue keeping the broadcasting times for the data items belonging to qw. After extracting data items in qw, the client removes all cells of $S_{ch}(l, k)$ from Q through subtraction. Then, the client downloads the extracted data items at the time dequeued from *dtQueue*.

- *Step* 3. In cases when Q is not empty, the client downloads regular data items belonging to qw in this step. To do this, the step begins with accessing $CI(l, k)$ with $l = 0$, i.e., *CI* of a cell on the top level, of the channel.
With $CI(l, k)$ accessed, the client enqueues the broadcasting time to access into *ctQuequ* using *ST* and *CT* of $CI(l, k)$, as follows.

◇ **getTimeWithST**.

For each entry $\langle k_s, t\rangle$ in ST,

$$\begin{cases} \text{if } (S_{ch}(l, k_s) \cap Q) \neq \phi \\ \text{if } t \neq \text{NULL} \quad \text{then } cpQueue \leftarrow t(\in ST) \\ \text{else subtract } S_{ch}(l, k_s) \quad \text{from } Q \\ \text{else } cpQueue \leftarrow t_r \end{cases}$$

◇ **getTimeWithCT**.

For each entry $\langle k_c, t\rangle$ in CT,

$$\begin{cases} \text{if } (S_{ch}(l, k_c) \cap Q) \neq \phi \\ \text{if } t \neq \text{NULL} \quad \text{then } cpQueue \leftarrow t(\in CT) \\ \text{else subtract } S_{ch}(l, k_c) \quad \text{from } Q. \end{cases}$$

After running getTimeWithST and getTimeWithCT, the client executes filterDataWithDT of Step 2 and then downloads all data items using *dtQueque*.

Algorithm 1 shows the procedure of window query processing with MLAIN.

4. Performance evaluation

For the performance evaluation, we implemented a testbed with the discrete time simulation package SimJava that consists of one broadcast server and 100 clients. The server disseminates the data items over a high-bandwidth downlink channel of 1 Mbps. The clients send their interested regions via a low-bandwidth uplink channel of 32 Kbps [19], where the interested regions follow Zipf distribution.

Algorithm 1 *WindowQuery*

Input: qw, a query window
Output: *Result*, a set of data items belonging to qw
1: $Q \leftarrow$ all cells on the bottom level overlapped with qw; // Step 1
2: *Index* $\leftarrow$ *HI* firstly encountered after tuning-in;
3: *ctQueue* $\leftarrow$ run getTimeWithHT;
4: **while** *ctQueue* is not empty **do** // Step 2
5: $t \leftarrow$ dequeue from *ctQueue*
6: *Index* $\leftarrow CI(l, k)$ accessed at t
7: *dtQueue* $\leftarrow$ run filterDataWithDT
8: *Result* $\leftarrow$ all data items accessed at the time dequeued from *dtQueue*;
9: **end while**;
10: *Index* $\leftarrow CI(l, k)$ with $l = 0$ firstly encountered; // Step 3
11: **while** Q is not empty **do**
12: *ctQueue* $\leftarrow$ run getTimeWithST;
13: *ctQueue* $\leftarrow$ run getTimeWithCT;
14: *dtQueue* $\leftarrow$ run filterDataWithDT;
15: *Result* $\leftarrow$ all data items accessed at the time dequeued from *dtQueue*;
16: $t \leftarrow$ dequeue from *ctQueue*
17: *Index* $\leftarrow CI(l, k)$ accessed at t
18: **end while**;
19: **return** *Result* ;

4.1. Simulation environments

We conducted simulations with a real dataset of 5922 Greek cities [20]. The parameters for the simulations are as follows: N is set to 5922; the size of one data item and one bucket, the smallest logical access unit, are 1024 and 64 bytes, respectively; the size of the cell identifier, the coordinates of a data item, and the time pointer are all set to 8 bytes; R_{qw}, the ratio of one side of the query window to one side of the data space D, is set from 0.01 to 0.2; H_r, the ratio of hot data items to all broadcast data items, is 0.1; N_q, the number of window queries issued by a client for one simulation condition, is 100,000; R_{hot}, the portion of accessing hot data items of N_q, is set from 0.5 to 1.0; m_c is set to 0.1% of N; and *freq* is set to 16, the same value as GDIN for the fair comparison. For each simulation condition, we measure the average access time and tuning time in buckets over N_q queries and the number of clients.

4.2. Comparison of the access time and tuning time

To show the effectiveness of MLAIN, we compare it with three existing flat broadcasting schemes, i.e., HCI, DSI, and CEDI, and one non-flat broadcasting scheme, GDIN, regarding access time and tuning time.

Fig. 4(a) shows the access time for various R_{hot} when R_{qw} is 0.07. As R_{hot} increases, i.e., the clients' data access pattern is more skewed to hot items, the access time of the two non-flat broadcasting GDIN and MLAIN decreases. This is a result of replicating hot items over a channel. In non-flat broadcasting, MLAIN outperforms GDIN because of the reduced number of data items repeated and the lessened differences between the indexes from applying a multi-leveled partition. The access time of MLAIN is about 89% of GDIN at $R_{hot} = 0.7$ and $R_{qw} = 0.07$.

Fig. 4(b) shows the access time for various R_{qw} when R_{hot} is 0.7. This figure shows that the proposed MLAIN also enables the clients to access hot items more rapidly than using other schemes, regardless of R_{qw} values when the clients' data patterns are skewed.

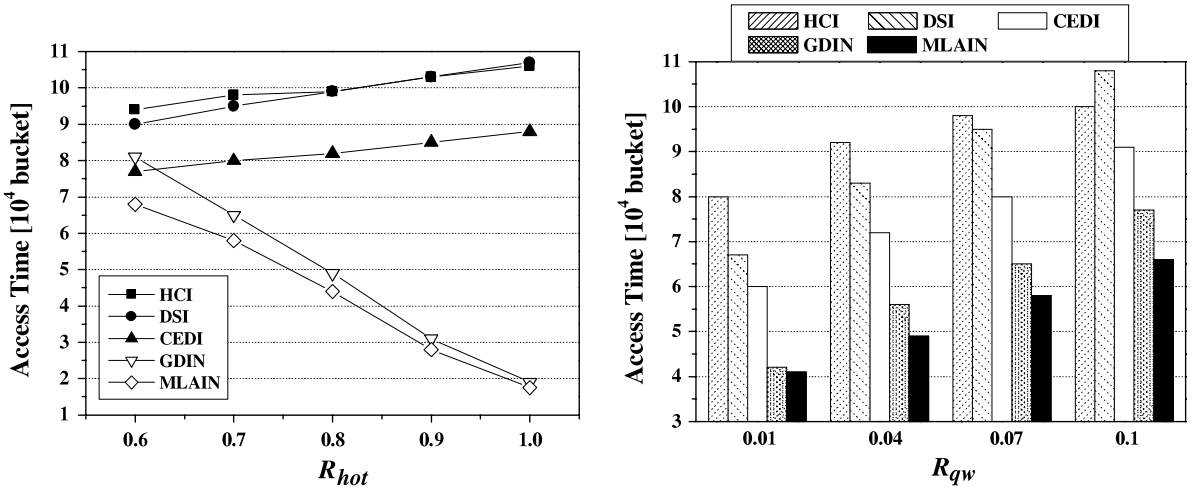
Fig. 5(a) compares the tuning time of MLAIN with the existing schemes. Observe that the tuning time of MLAIN is much shorter than that of the DSI and HCI and is almost same as CEDI and GDIN. This is because MLAIN, GDIN, and CEDI allow the clients to access only the targeting data items within the given query window qw by extracting them using their original coordinates before accessing them, while the HCI and DSI make clients extract them after making accesses filter out excessive candidate data items using HC values. The tuning time of MLAIN is about 98% of GDIN at $R_{hot} = 0.7$ and $R_{qw} = 0.07$.

Fig. 5(b) shows the tuning time of MLAIN and other existing schemes for various R_{qw} when R_{hot} is 0.7. MLAIN has almost the same tuning time as GDIN and CEDI, but is shorter than that of the HCI and DSI due to the reasons mentioned above.

4.3. Practical implications

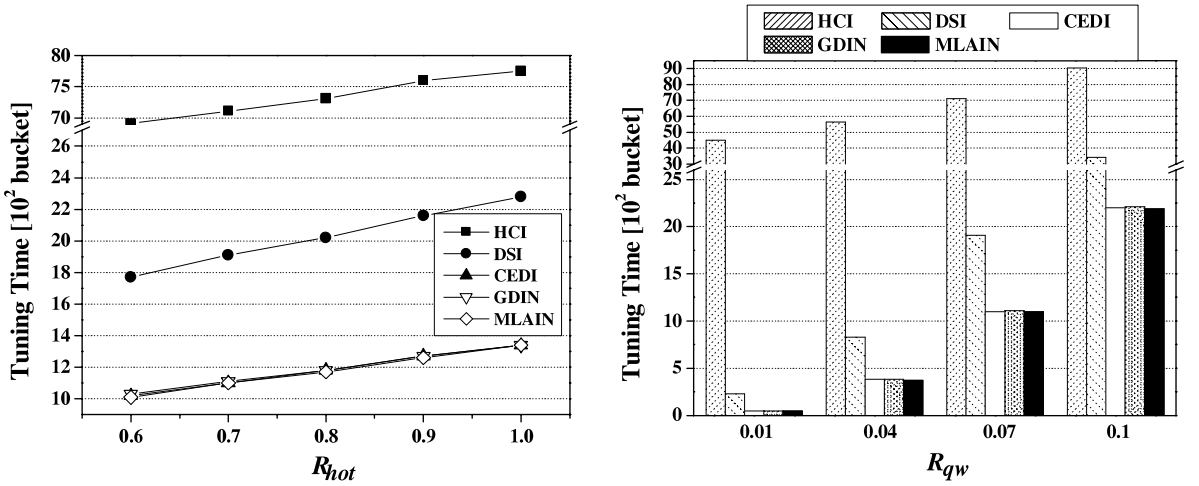
In this section, we show the practicality of the proposed MLAIN using the amount of the average energy, E_q, consumed during query processing. We can compute E_q with Eq. (4) below using the real energy parameters.

$$E_q = \varepsilon_{active} \times T_{tuning} + \varepsilon_{doze} \times (T_{access} - T_{tuning}). \quad (4)$$



(a) R_{hot}. (b) *Window ratio.*

Fig. 4. Comparison of MLAIN with other schemes for access time.



(a) R_{hot}. (b) *Window ratio.*

Fig. 5. Comparison of MLAIN with other schemes regarding tuning time.

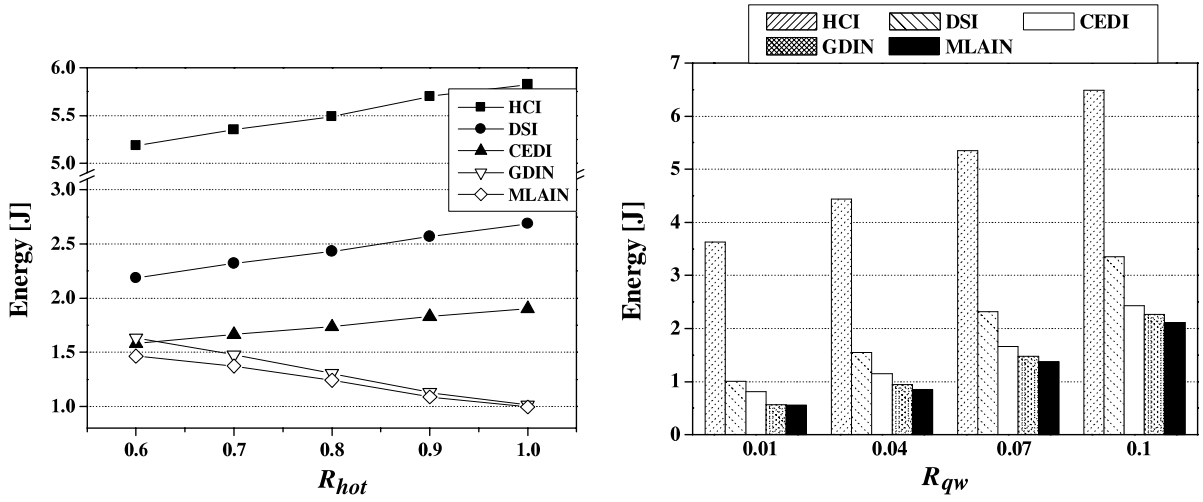
Table 1
Energy consumption of the client (in mW/s).

	Model	ε_{doze}	ε_{active}
CPU	StrongARM SA-1100	0.16	400
NIC	RangeLAN2 7401/02	25	750

Here, ε_{active} is the amount of energy consumed in active mode per second and ε_{doze} in doze mode. T_{tuning} and T_{access} are the average tuning and access time in seconds, respectively, which are transformed from in bucket to in second using the bandwidth 1 Mbps of the downlink channel. The first term and second term of Eq. (4) mean the amount of the average energy consumed in active mode and doze mode, respectively. In the second term, $(T_{access} - T_{tuning})$ means the average length of time that clients stay in doze mode while processing N_q queries.

For real energy parameters, we take into consideration two components of the clients, i.e., the CPU and network interface card (NIC), operating in active mode and doze mode, respectively [21]. Table 1 shows the values of ε_{active} and ε_{doze} consumed by the CPU and NIC. Thus, each client consumes 25.16 mW/s in doze mode and 1150 mW/s in active mode [22].

Fig. 6(a) shows the energy consumption of the clients for various R_{hot} when R_{qw} is 0.07. It reveals that the proposed MLAIN requires lower energy consumption than the other schemes in skewed data access. At $R_{hot} = 0.7$ and $R_{qw} = 0.07$, E_q by MLAIN is about 25%, 59%, 82%, and 92% of that by the HCI, DSI, CEDI, and GDIN, respectively. This is because MLAIN has a



(a) R_{hot}. (b) *Window ratio.*

Fig. 6. Comparison of MLAIN with other schemes regarding energy consumption.

shorter access time and tuning time than the others. It also means that MLAIN makes it possible for clients to process more queries than the other schemes using the same amount of energy. The figure shows that how practical the proposed MLAIN is over the existing schemes when the clients' data access pattern is skewed. Fig. 6(b) shows the energy consumption for various R_{qw} when R_{hot} is 0.7. MLAIN outperforms the others for the reasons mentioned above.

5. Conclusion

In this paper, we have addressed the problem of processing window queries efficiently in non-flat data broadcasting. We proposed MLAIN, which allows clients to access hot data items quickly when their data access patterns are skewed to hot data items. MLAIN partitions the data space by recursively subdividing it into four quadrant cells of multiple levels, with the constraint restricting the number of data items in a cell. The constraint makes MLAIN different from other indexing schemes that use a regular space partition. It reduces the number of data items in a hot cell when lots of regular data items are gathered around hot data items. The reduced number of data items shortens the broadcast cycle and the spacing between indexes on the channel, resulting in quick data access. The cell indexes and hot cell indexes of MLAIN provide link information between cells as well as multiple paths to a cell to help clients access targeting cells efficiently. Performance evaluation by simulation shows the proposed MLAIN outperforms the other existing schemes in access time, tuning time, and energy consumption. For future study, we are investigating k nearest neighbors query over non-flat broadcasting.

References

[1] Shih-Hao Hung, Chi-Sheng Shih, Jeng-Peng Shieh, Chen-Pang Lee, Yi-Hsiang Huang, Executing mobile applications on the cloud: framework and issues, Computers and Mathematics with Applications 63 (2) (2012) 573–587.
[2] Jameela Al-Jaroodi, Imad Jawhar, Alyaziyah Al-Dhaheri, Fatmah Al-Abdouli, Nader Mohamed, Security middleware approaches and issues for ubiquitous applications, Computers and Mathematics with Applications 60 (2) (2010) 187–197.
[3] Zhuzhong Qian, Ce Chen, Ilsun You, Sanglu Lu, ACSP: a novel security protocol against counting attack for UHF RFID systems, Computers and Mathematics with Applications 63 (2) (2012) 492–500.
[4] Hoon Ko, Kyungjin An, Taihoon Kim, Goreti Marreiros, Zita Vale, Jongmyoung Choi, A study on dynamic state information (DSI) around users for safe urban life, Computers and Mathematics with Applications 63 (2) (2012) 554–563.
[5] T. Imielinski, S. Viswanathan, B.R. Bardrinath, Data on air: organization and access, IEEE Transactions on Knowledge and Data Engineering 9 (3) (1997) 353–372.
[6] Wang-Chein Lee, Baihua Zheng, DSI: a fully distributed spatial index for location-based wireless broadcast services, in: Distributed Computing Systems, 2005, ICDCS05, 25th IEEE International Conference on, 2005, pp. 349–358.
[7] SeokJin Im, MoonBae Song, Jongwan Kim, Sang-Won Kang, Chong-Sun Hwang, An error-resilient cell-based distributed index for location-based wireless broadcast services, in: Data Engineering for Wireless and Mobile Access Workshops, 2006, MobiDE06, Fifth International ACM Workshop on, pp. 59–66.
[8] SeokJin Im, Hee Yong Youn, JinTak Choi, Jinsong Ouyang, A novel air indexing scheme for window query in non-flat wireless spatial data broadcast, Journal of Communications and Networks 13 (4) (2011) 400–407.
[9] Demin Li, Qiuran Li, Jiacun Wang, Zhitao Zhang, Navigation function design for backbone connectivity in vehicle ad hoc networks, Computers and Mathematics with Applications 61 (8) (2011) 2067–2070.
[10] SeokJin Im, Hee Yong Youn, A cell-based hybrid indexing scheme for energy conserving k nearest neighbor search on air, IEICE Transactions on Communications E91-B (11) (2008) 3799–3802.
[11] HaRim Jung, Yon Dohn Chung, Ling Liu, Processing generalized k-nearest neighbor queries on a wireless broadcast stream, Information Sciences 188 (1) (2012) 64–79.
[12] URL: http://www.microsoft.com/industry/government/federal/do-ddirectband.mspx.

[13] S. Acharya, R. Alonso, M. Franklin, S. Zdonik, Broadcast disks: data management for asymmetric communications environments, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD'95, 1995, pp. 199–210.
[14] Baihua Zheng, Wang-Chien Lee, Dik Lun Lee, Spatial queries in wireless broadcast systems, Wireless Network 10 (6) (2004) 723–736.
[15] Ken C.K. Lee, Brandon Unger, Baihua Zheng, Wang-Chien Lee, Location-dependent spatial query containment, Data and Knowledge Engineering 70 (10) (2011) 842–865.
[16] N.H. Vaidya, S. Hameed, Scheduling data broadcast in asymmetric communication environments, Wireless Networks 5 (3) (1999) 171–182.
[17] J. Xu, X. Tang, Wang-Chien Lee, Time-critical on-demand data broadcast: algorithms, analysis, and performance evaluation, IEEE Transactions on Parallel and Distributed Systems 17 (1) (2006) 3–14.
[18] Yuxia Yao, Xueyan Tang, Ee-Peng Lim, Aixin Sun, An energy-efficient and access latency optimized indexing scheme for wireless data broadcast, IEEE Transactions on Knowledge and Data Engineering 18 (8) (2006) 1111–1124.
[19] S. Acharya, S. Muthukrishnan, Scheduling on-demand broadcasts: new metrics and algorithms, in: MobiCom99: Proceedings of the 4th ACM International Conference on Mobile Computing and Networking, ACM, 1999, pp. 43–54.
[20] Real datasets. URL: http://www.rtreeportal.org.
[21] SeokJin Im, MoonBae Song, Sang-Won Kang, Jongwan Kim, Chong-Sun Hwang, Sangkeun Lee, Energy conserving multiple data access in wireless data broadcast environments, IEICE Transactions on Communications E90-B (9) (2007) 2629–2633.
[22] O. Kasten, Energy Consumption, ETH-Zurich, Swiss Federal Institute of Technology.
URL: http://www.inf.ethz.ch/personal/kasten/research/bathtub/energy_consumption.htm.