EFFICIENT PROCESSING OF SPATIAL RANGE QUERIES ON WIRELESS BROADCAST STREAMS

KwanHo In, HaRim Jung, Hee Yong Youn and Ung-Mo Kim

School of Information and Communication Engineering,
Sungkyunkwan University, Suwon, Korea

ABSTRACT

With advances in wireless networks and hand-held computing devices equipped with location sensing capability (e.g., PDAs, laptops, and smart phones), a large number of location based services (LBSs) have been successfully deployed. In LBSs, wireless broadcast is an efficient method to support the large number of users. In wireless broadcast environment, existing research proposed to support range queries search, may tune into unnecessary indexes or data object. This paper addresses the problem of processing range queries on wireless broadcast streams. In order to support range queries efficiently, we propose a novel indexing scheme called Distributed Space-Partitioning Index (DSPI). DSPI consists of hierarchical grids that provide mobile clients with the global view as well as the local view of the broadcast data. The algorithm for processing range queries based on DSPI is also proposed. Simulation experiments demonstrate DSPI is superior to the existing index schemes.

KEYWORDS

Location dependent information services, continuous range queries, moving objects, index structures, broadcast systems

1. INTRODUCTION

With advances in wireless technologies and the wide spread of mobile devices, the trend toward pervasive computing has gained momentum. Since location (e.g., 2-dimensional coordinates) is one of the most important properties in a pervasive computing environment, various location dependent information services (LDISs) have emerged as one of the promising applications [3-4, 10-11]. In order to support LDISs, efficient processing of location dependent queries (LDQs), which retrieve information based on the current location of mobile clients (MCs), is critical. One of the essential classes of LDQs is the range query which finds out all data objects within a given rectangular (or circular) region.

Wireless data broadcast is considered to be an effective way for disseminating location dependant data (LDD) and supporting LDQs since it leverages computational capability of MCs, and thus accommodates a huge number of MCs simultaneously. In this paper, we address the problem of processing range queries (One of the essential classes of LDQs) in wireless data broadcast environments.

Preliminaries. In wireless data broadcast, data (e.g., LDD) are periodically broadcasted. The server periodically broadcasts data through public wireless channels. MCs then retrieve the data to evaluate their queries on the broadcast stream, without sending any request to the server. An important characteristic of wireless data broadcast is that data objects are sequentially delivered on the air and available to MCs only when they are currently being broadcasted. In order to retrieve the data necessary for query processing, MCs have to continuously listen to the broadcast

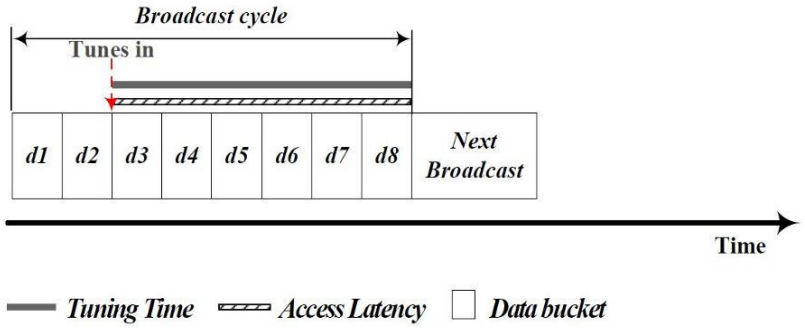
DOI : 10.5121/ijdms.2014.6103

channel until the data arrive. This leads to vast energy consumption on MCs. Current mobile devices usually support two operation modes: full operational mode called active mode and energy conserving mode referred to as doze mode. MCs can conserve their energy if they can stay in doze mode most of the time and switch to active mode only when the desired data object arrives on the air.

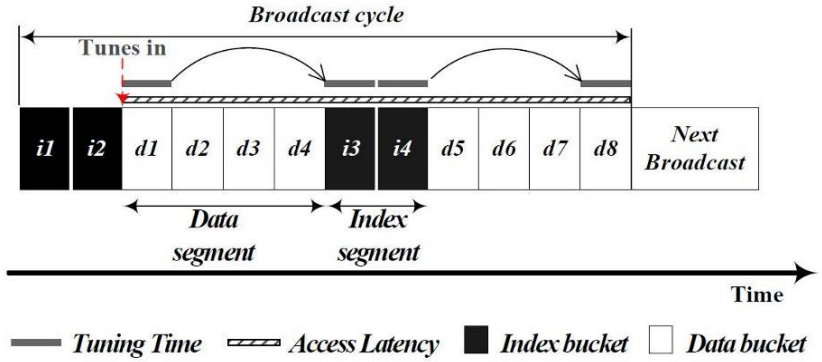
Air indexing is commonly used to help MCs predict the arrival time of the desired data. The basic idea is to interleave index information (e.g., arrival times of data) with data on the broadcast stream. By first accessing the index information, MCs can selectively retrieve only the desired data. However, the average waiting time of the MCs is increased because the insertion of index information extends the broadcast cycle (e.g., the length of the wireless broadcast stream). In wireless data broadcast, the following measures are commonly used as performance criteria [1]:

● *Access Latency*: The time elapsed from the moment an MC requests data to the moment all the required data are retrieved by the MC. This corresponds to the waiting time of the MC.

●*Tuning Time*: The total amount of time spent by the MC in the active mode for actually listening to the broadcast channel. The Tuning Time is commonly used for measuring the energy-efficiency of wireless broadcast data access.



(a) Broadcast stream without an index.



(b) Broadcast stream with an index

Figure 1. Data retrieval without and with an index

Both Access Latency and Tuning Time are evaluated in terms of the number of buckets, where a bucket is the smallest logical unit of wireless data broadcast. The broadcast stream consists of index buckets and data buckets. Index buckets hold the index information, while data buckets

hold the data. A set of contiguous index buckets is referred to as an index segment and a set of contiguous data buckets broadcasted between successive index segments is called a data segment.

Let us consider a server that periodically broadcasts dataset D, where $D = \{d1, d2, d3, \cdots, d8\}$, where $di(1 \leq i \leq 8)$ is a data object. Suppose that an MC wants to retrieve d8. Fig. 1(a) and Fig. 1(b) illustrate the process of data retrieval on the wireless broadcast stream without and with an index, respectively. Let us assume that the MC tunes into the broadcast channel as depicted in the figures.

Without an index, the MC has to continually listen to the broadcast channel until it retrieves $d8$ as shown in Fig. 1(a). On the contrary, as illustrated in Fig. 1(b), the MC can stay in doze mode and only wake up when $d8$ arrives after accessing index (e.g., $i3$ and $i4$). Note that each bucket contains a temporal offset to the beginning of the next index segment to facilitate the access of index. Although the MC reduces the Tuning Time with the help of index, it suffers from the Access Latency due to the lengthened broadcast cycle. In order to support efficient query processing in wireless data broadcast, the air index scheme has to minimize Tuning Time while minimizing Access Latency by keeping the increase of the broadcast cycle minimal.

Contributions. In this paper, we propose a novel air index structure, called Distributed Space Partitioning Index (DSPI), for supporting range queries in wireless data broadcast environments. DSPI consists of a hierarchy of grids, each of which uniformly partitions the data space with different granularity. Additionally, DSPI has a linear, and yet distributed structure suitable for wireless data broadcast. The technical contributions of the DSPI are summarized as follows:

- Providing efficient guidance for MCs to selectively retrieve only the required data objects by letting the MCs traverse a hierarchy of grids.
- Enabling MCs to quickly process range queries by reducing the size of index.
- Having a linear, and yet distributed structure suitable for sequential data delivery in wireless data broadcast while preserving the spatial locality of data objects.

Organization. The rest of the paper is organized as follows. In Section 2, related work to our study is presented. Section 3 presents the proposed index scheme, namely DSPI. In Section 4, experiments are conducted to show the efficiency of DSPI. Finally, we conclude this paper in Section 5.

2. RELATED WORKS

In mobile computing environments, wireless broadcasting has been widely adopted for delivering information to a large number of users due to its scalability benefit [12-15]. In a wireless broadcasting system, the server periodically broadcasts data objects through the wireless broadcasting channel in a pre-scheduled sequential order. When each mobile device receives a query from a user, it tunes into the channel and processes the query on the broadcast stream.

Air index interleaving [16] is commonly used for reducing the tuning time at the expense of the increased access time. The basic idea is to interleave an index information with data objects on the broadcast stream. By first examining the index information, the mobile client can get the arrival times of the relevant data objects for the given query. A wireless data broadcast, the concept of air indexing has received much attention and was first discussed in [1]. In [1], the authors proposed the (1, m) *indexing* and the *distributed indexing schemes*. Both schemes apply an *index tree* (e.g., B^+-tree), where each node stores the ids and arrival times of its children. In order to start query processing, an MC should wait until the root node of the index tree arrives.

Fig. 2 shows an example of the index tree for a broadcast stream. Circles represent index nodes and the rectangles in the bottom represent data objects.

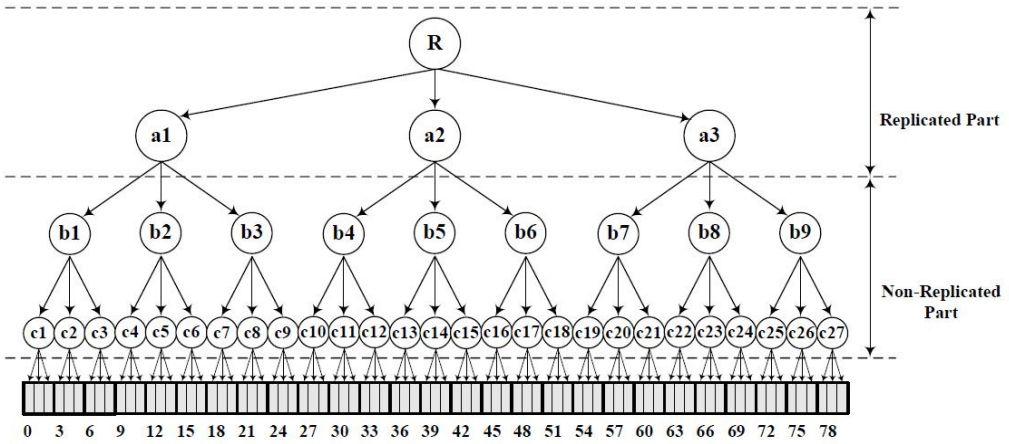


Figure 2. Example of the index tree

In the (1, m) indexing, the entire index is replicated m times and inserted into every 1/m fraction of the broadcast stream (See Fig. 3). By replicating the entire index for m times, the waiting time for reaching the root node can be reduced. This, however, incurs a huge amount of Access Latency since the duplication of the entire index lengthens the broadcast cycle.

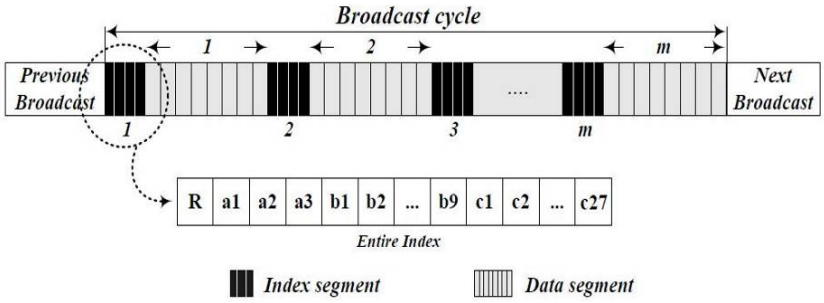
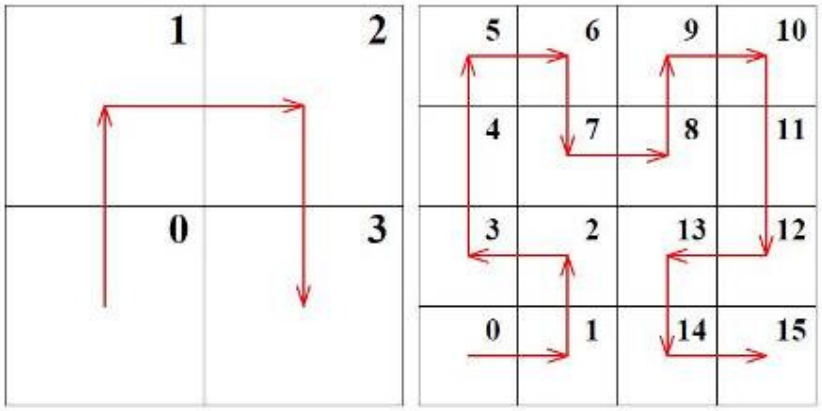


Figure 3. Example of (1, m) indexing

In the distributed indexing, an index is partially replicated and only the relevant portion of the index is inserted in front of the data segment which immediately follows it. Since the distributed indexing scheme replicates only the upper level of the index tree (*Replicated Part* in Fig. 2), the size of broadcast stream is small compared with the that of (1, m) indexing. As a result, distributed indexing shows better performance than the (1, m) indexing in terms of Access Latency. Both (1, m) indexing and distributed indexing focus only on the id-based data retrieval. That is, data objects are retrieved based on their identifiers. To process LDQs on the wireless data broadcast stream, an indexing scheme should support location-based data retrieval. Recently, several index structures for supporting LDQs in wireless data broadcast have been proposed [4, 5, 7, 8]. Among them, we give a brief overview of the latest air indexing schemes, namely *Hilbert Curve Index (HCI)* [7] and *Distributed Spatial Index (DSI)* [8].

To fit the sequential nature of wireless data broadcast, HCI and DSI utilize the *Hilbert curve* (HC), one of the well-known *space-filling curves*. The Hilbert curve is a continuous path which visits all cells in a multi-dimensional grid exactly once without crossing itself. Fig. 4(a) and 4(b)

show an example of Hilbert curves of order 1 and 2, for 21×21 grid and 22×22 grid respectively. Numerical values in the figures represent the visiting order of the curves.



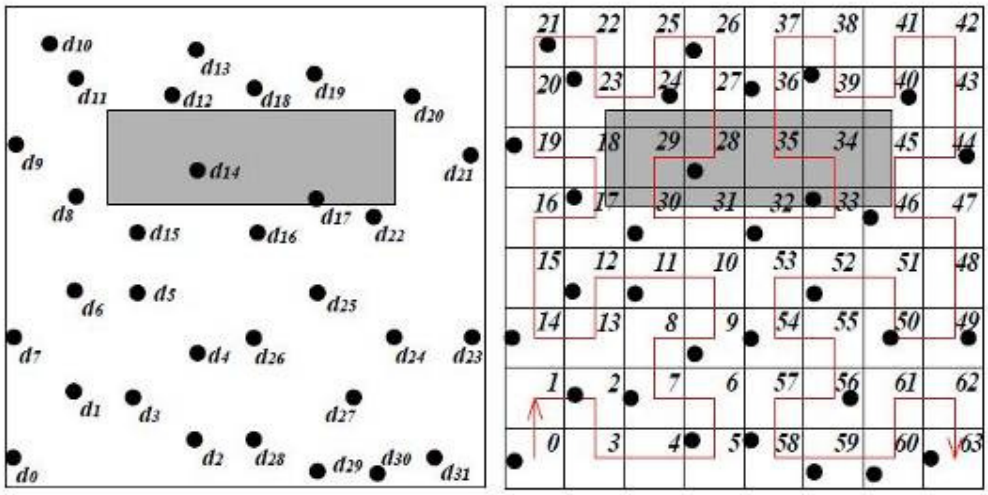
(a) Order 1 (b) Order 2

Figure 4. Hilbert curves of order 1 and 2.

Since the Hilbert curve can efficiently map 2-dimensional space to 1-dimensional space without losing the locality of the underlying data, it facilitates the linear scan of the 2-dimensional data delivered through the wireless broadcast channel. Specifically, the location (e.g., coordinates) of each data object is approximated by a unique integer called HC value. This is done by partitioning the data space into $2^{\lambda}\times2^{\lambda}$ grid cells, so that each cell contains only one data object. Here, λ is the order of Hilbert curve. Then each cell is assigned a HC value according to the visiting order of Hilbert curve. Finally, the location of the data object inside each cell is approximated by the corresponding HC value. Fig. 5(a) and 5(b) illustrate a sample dataset containing 32 data objects and their approximated locations respectively. For example, the locations of data objects $d1$ and $d2$ are approximated by 2 and 5.

In HCI and DSI, data objects are broadcast in the increasing order of their HC values. HCI constructs a B^{+}-tree, where leaf nodes contain the HC values of all the data objects together with corresponding pointers. Each data object is indexed and identified by its HC value (e.g., approximated location), and the associated pointer indicates the arrival time of the data object. To reduce the waiting time for reaching the root node of the index, HCI utilizes the (1, m) indexing scheme.

DSI, a table-based indexing scheme, organizes the index structure in a fully distributed fashion. The basic idea of DSI is to split the broadcast stream into equal-sized frames, each of which contains (i) a number of data objects and (ii) index table. An index table consists of $\log_r N_F$ entries, where r is the exponential base and NF is the number of frames in a broadcast cycle. The ith entry of any index table is represented as a $< HC_i, ptr_i >$ tuple. Here, HC_i is the smallest HC value of the data objects within the frame pointed by ptr_i, and ptr_i is the pointer which indicates the arrival time of the next r^ith frame. In this manner, DSI enables each frame to maintain the indexing information of the data objects to be broadcast.



(a)Dataset (b)HC approximation (λ=3)

Figure 5. Dataset and Hilbert curve approximation

With both HCI and DSI, an MC has to retrieve data objects more than necessary to process a range query, because the locations of data objects are approximated by the HC values. For example, consider the processing of the range query *RQ* (shaded rectangle) in Fig. 5 (b). With the HCI, the MC should retrieve all data objects, whose HC value ∈ {17, 23, 24, 28, 30, 32, 33, 36, 39, 40, 46}. However, only the data objects, whose HC value ∈ {28, 33}, can be the result of RQ. Although DSI reduces the number of data objects being retrieved, it cannot completely eliminate the unnecessary retrieval of data objects. This increases *Tuning Time* of the MC. Furthermore, not only the distribution of LDD in the original space but also the number of LDD may greatly deviate the performance of both HCI and DSI in terms of *Access Latency* as well as *Tuning Time*.

The bucket is the basic unit of wireless broadcasting, and the broadcast stream consists of *index buckets* and *data buckets*. Index buckets hold index information, whereas data buckets hold data. We assume that each bucket has the same capacity, irrespective of its type (i.e., index bucket or data bucket). In order to make all buckets self-identifying, each bucket contains the following header information: bucket id, bucket type, upcoming appearance time of the next index bucket in the broadcast stream, and start time of the next broadcast stream. Note that the bucket id is determined by the offset of the corresponding bucket from the start time of the current broadcast stream [17]. In the remainder of this paper, we use the number of buckets to measure access time and tuning time. The number of buckets can be translated into time values without loss of generality because all buckets are assumed to have the same capacity and the bandwidth of the broadcasting channel is fixed.

3. DISTRIBUTED SPACE-PARTITIONING INDEX (DSPI)

In this section, we propose Distributed Space-Partitioning Index (DSPI) that enables MCs to selectively retrieve only the required data objects (e.g., LDD). The proposed indexing scheme significantly reduces *Tuning Time* and *Access Latency* compared to the previous indexing schemes in [3, 4].

3.1. Index Structure

DSPI consists of n_G hierarchical levels of grids, where n_G is the number of grids. Each level of grids partitions the data space with different granularity into grid cells of the same size. Fig. 6 shows the basic idea of DSPI, when $n_G = 2$. For simplicity, we assume that $n_G = 2$ in the rest of the paper. However, extension to $n_G > 2$ is straight forward.

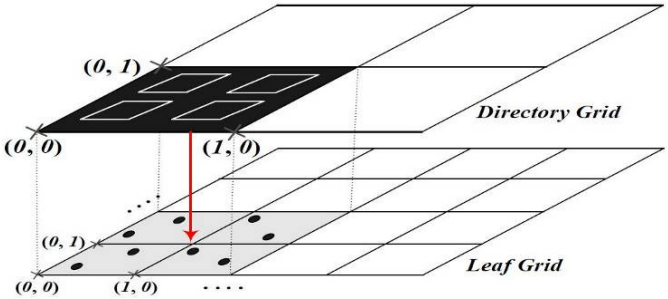


Figure 6. Hierarchical grids

The lowest level of the grids, referred to as the leaf grid, indexes the underlying dataset, whereas the upper level of the grids, called the *directory grid*, indexes the leaf grid. For the linear placement of the grid cells as well as data objects on the air, we use the Hilbert curve since it preserves the spatial locality better than other curves. Let DS be a two-dimensional data space. For simplicity, we assume that DS is the unit square $(0, 1)^2$ in our discussion.

Definition 1: (Leaf grid). The leaf grid is a grid that uniformly partitions the DS into $2^{\gamma L} \times 2^{\gamma L}$ grid cells, where γL (≥ 1) is the granularity factor of the leaf grid. We call a grid cell in the leaf grid a leaf cell.

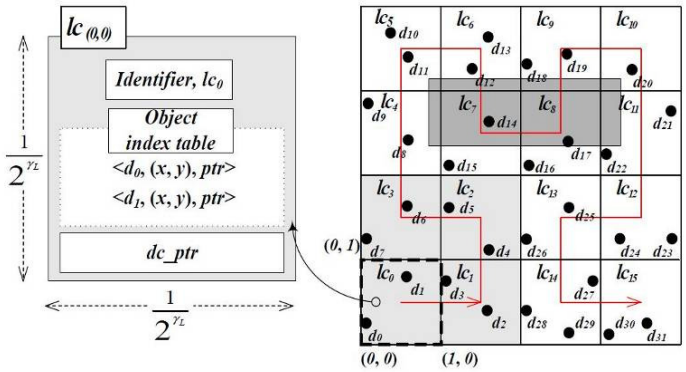


Figure 7. Leaf grid (γL=2)

The leaf grid has the finest granularity among the grids in DSPI. It provides MCs with local view of the underlying dataset D. The extent of a leaf cell on every dimension is $1/2^{\gamma L}$, so that the leaf cell $lc_{(i,j)}$ at column i and row j (starting from the low-left corner of the leaf grid) maintains the detailed information of all data objects with x-coordinate in the range $(i \times 1/2^{\gamma L}, (i+1) \times 1/2^{\gamma L})$ and y-coordinate in the range $(j \times 1/2^{\gamma L}, (j+1) \times 1/2^{\gamma L})$. Specifically, each leaf cell maintains the following information:

●**Identifier**: A leaf cell is assigned a HC value as an identifier according to the occurring order on the Hilbert curve of order $\lambda = \gamma L$. The identifier determines the broadcast order of the leaf cell.

●**Object index table**: An object index table stores identifiers, locations and arrival times of all data objects which belong to the leaf cell. It consists of a set of tuples in the form of <*id*, (*x, y*), *ptr*>.

●**Directory cell pointer *dc_ptr***: *dc_ptr* indicates the arrival time of the next directory cell.

The leaf grid enables MCs to retrieve only the required data objects for range query processing by using the 2-dimensional coordinates, instead of approximated coordinates, of data objects. Figure 7 illustrates the structure of the leaf grid based on the sample dataset in Fig. 5(a). In the broadcast stream, all the leaf cells are sorted according to their identifiers (e.g., HC values), and each leaf cell is placed in front of the relevant data segment containing data objects belonging to the leaf cell.

Definition 2: (Directory grid). The directory grid is a grid that uniformly partitions the DS into $2^{\gamma D} \times 2^{\gamma D}$ grid cells, where γD ($1 \leq \gamma D < \gamma L$) is the granularity factor of the directory grid. We call a grid cell in a directory grid a directory cell. In particular, the directory grid which has the coarsest granularity is called the root grid.

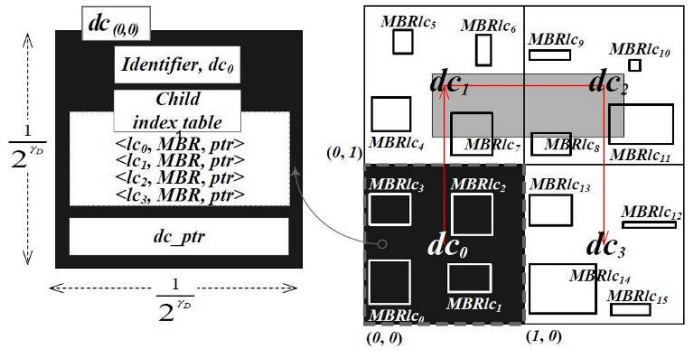


Figure 8. Directory grid ($\gamma D = 1$)

Since we assume $n_G = 2$, hereafter we use directory grid and root grid interchangeably. Directory grid maintains information of the lower level grid, namely, the leaf grid (Note that we assume $n_G = 2$) as well as the global view of D. The extent of each directory cell on every dimension is $1/2^{\gamma D}$. The directory cell $dc_{(i,j)}$ maintains the information of its child cells, i.e., all leaf cells inclosed by the range $(i \times 1/2^{\gamma D}, (i+1) \times 1/2^{\gamma D}) \times (j \times 1/2^{\gamma D}, (j+1) \times 1/2^{\gamma D})$. For example, the directory cell $dc_{(0,0)}$ in Fig. 8 contains the information of the child cells $lc_{(0,0)}$, $lc_{(1,0)}$, $lc_{(0,1)}$ and $lc_{(1,1)}$ in Fig. 7. Each directory cell includes the following information:

●**Identifier**: A directory cell is assigned an HC value as an identifier according to the occurring order on the Hilbert curve of order $\lambda = \gamma D$. The identifier determines the broadcast order of the directory cell.

●**Child index table**: A child index table stores the identifier, the minimum bounding rectangle (MBR) and the arrival time of each child cell. The child index table consists of a number of tuples in the form of <*id*, MBR, *ptr*>. The MBR is a rectangle of minimum area that fully encloses all the data objects belonging to the corresponding child cell.

●**Directory cell pointer *dc_ptr***: *dc_ptr* indicates the arrival time of the next directory cell.

The directory grid enables MCs to selectively read only the necessary portion of the leaf grid, by examining the MBRs of its child cells. Fig. 8 illustrates the structure of the directory grid based

on the leaf grid in Fig. 7. All the directory cells are sorted according to their identifiers and each of them is placed in front of its child cells on the broadcast stream. Fig. 9 illustrates the wireless broadcast stream generated from the hierarchical grids shown in Fig. 7 and 8.

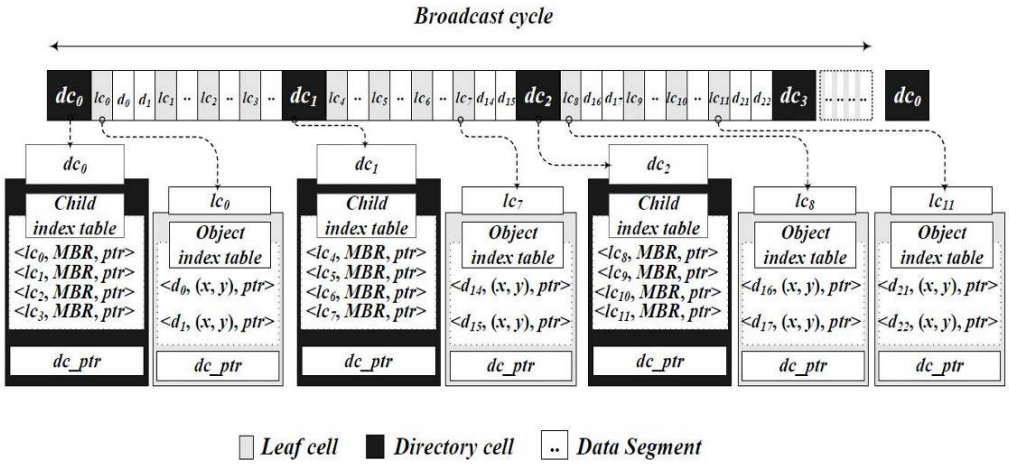


Figure 9. Broadcast stream generated from the hierarchical grids

3.2. Range Query Processing

A range query retrieves all data objects within a given rectangular (or circular) region. More formally, given a set of 2-dimensional dataset D= $\{d_1, d_2, \ldots, d_{|D|}\}$and a query region q, the result of range query is R(q)= $\{d \in D \mid d \text{ is within } q\}$. In our discussion, the query region q is represented by a rectangle.

To process a range query using DSPI, an MC first detects a collection of directory cells which overlap with the query region q, and thereafter it retrieves the qualified data objects by traversing the directory cells and their child cells on the broadcast stream.

In particular, given a set of overlapped directory cells, an MC maintains a list L of their identifiers (ids). Note that given the mapping function of the Hilbert curve, it is easy for an MC to determine the ids (HC values) of the overlapped directory cells. Then, the MC performs *initial probe*, i.e., tuning into the broadcast channel to find out when the next directory cell arrives on the broadcast stream. After the initial probe, the MC continues to perform the following steps until L is empty:

1. The MC sequentially accesses a number of directory cells until it reaches the closest directory cell *dc* on the broadcast stream whose id$\in L$. This is done by following *dc_ptr* in each accessed directory cell. Then, the MC removes the id of dc from L.

2. Using the child index table in *dc*, the MC determines (i) the child cells whose MBRs intersect with the query region q and (ii) when to tune into the broadcast channel in order to access them.

3. The MC selectively accesses each qualified child cell. For each qualified child cell *lc*, the MC determines (i) the data objects which are within the query region q and (ii) the arrival times of them by using the object index table in *lc*. Then, the MC selectively retrieves the qualified data objects.

Example. Consider the range query with its query region q (shaded rectangle) either in Fig. 7 or 8. As the first step, the MC determines a set of overlapped directory cells, namely $dc_{(0,1)}$ and $dc_{(1,1)}$, from the directory grid in Fig. 8 and inserts their ids (dc_1 and dc_2) into the list *L*. Then, the MC tunes into the broadcast channel to find out when the next directory cell will arrive on the broadcast stream.

Without loss of generality, we assume that the MC first tunes into the broadcast channel at the time (or position) of dc_1, which contains the information of its child cells (lc_4, … , lc_7) as well as their MBRs. Since $dc_1 \in$ L, the MC removes dc_1 from L and determines the child cells whose MBRs intersect with the query region *q*. Because only the MBR of lc_7 overlaps with the *q* as shown in Fig. 8, the MC accesses only lc_7 and selectively retrieves the qualified data object, namely d_{14}, by utilizing the object index table in lc_7. The MC then accesses to the next closest directory cell on the broadcast stream whose id$\in L$, namely dc_2, by using the *dc_ptr* in dc_1. Then, the MC removes dc_2 from *L*. Since MBRs of lc_8 and lc_{11} overlap with the query region *q* (See Fig. 8), the MC accesses both lc_8 and lc_{11}, and then retrieves d_{17}. In this way, the MC completes the processing of range query. Algorithm 1 presents the detailed algorithm for processing range queries based on DSPI.

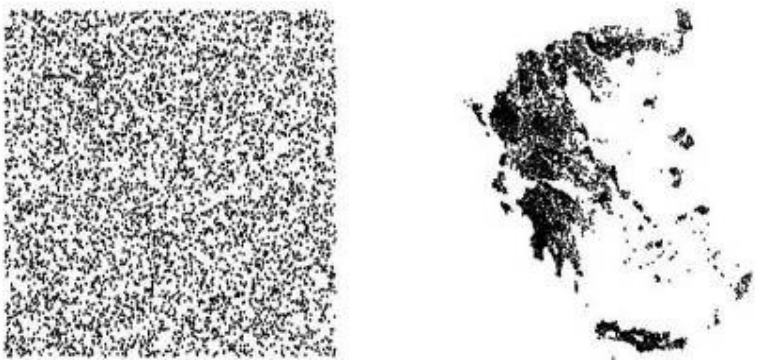
Algorithm 1 Range Query Processing

```
Input. q: a query rectangle
       L: a list containing the ids of overlapped directory cells
Output. data objects within q
Procedure.
1:  QueryResult=∅;
2:  perform initial probe;
3:  do{
4:      access the closest directory cell dc whose id∈L;
5:      remove id of dc from L;
6:      for each child cell lc belonging to dc do
7:          if( MBR overlaps with q) then
8:              for each data object d belonging to lc do
9:                  if (d is within q) then
10:                     QueryResult = QueryResult ∪ {d};
11:                 end if
12:             end for
13:         end if
14:     end for
15: }while (L is not empty)
16: return QueryResult;
```

4. PERFORMANCE EVALUATION

We evaluated the performance of DSPI by comparing its *Access Latency* and *Tuning Time* with those of HCI and DSI. For the implementation of the HCI, the (1, m) indexing scheme was utilized. We used the optimal *m* (=$\sqrt{(DATA_{size}/INDEX_{size})}$) to minimize Access Latency, where $DATA_{size}$ and $INDEX_{size}$ are the sizes of whole data objects and the global index in terms of the number of buckets [1]. On the other hand, for the DSI, the exponential base *r* was set to 2 and the number of data objects within each frame was determined so that the index table associated with a frame could fit into one bucket [8]. For the configuration of DSPI, we set n_G = 2. In addition, the granularity factor γL of the *Leaf Grid* was set to 4, and the granularity factor γ_D of *Directory Grid* was set to 3. The system model, which consists of a broadcast server, MCs and a broadcast channel, was implemented in the Java language.

We conducted performance analysis on two datasets (See Fig. 10): UNIFORM dataset and REAL dataset. In the UNIFORM dataset, 6,000 data objects are uniformly generated in a square Euclidian space. The REAL dataset contains 5,848 actual cities and villages of Greece, which is extracted from the dataset available form [9]. In the experiment, 10,000 queries were issued. We defined the ratio of average side length of a query rectangle to that of the whole data space, referred to as *RectSideRatio*. We used the number of bytes instead of the number of buckets to evaluate *Access Latency* and *Tuning Time* because the bucket size is varied in the experiment [8]. The number of bytes can be translated into the time values without loss of generality, since the bandwidth of wireless channel is fixed. The size of a data object is 1024 bytes, and 16 bytes are used for representing the 2-dimensional coordinates of a data object (8 bytes for each coordinate). 16 bytes are also used for representing the HC value. We set the pointer size to 2 bytes. Table 1 illustrates parameter settings used in the experiments. Since the number of MCs does not affect the system performance [7], we considered only one MC in the performance evaluation.



(a)UNIFORM (b)REAL

Figure 10. Datasets for performance evaluation

Table 1. Parameter Settings

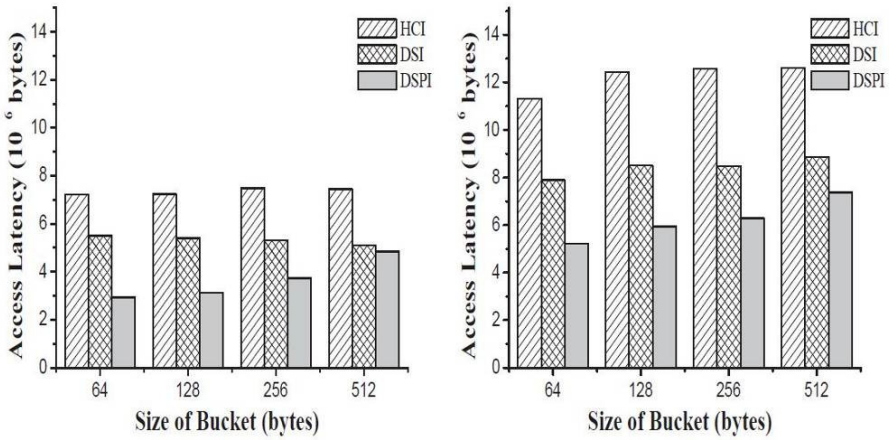
Parameter	Setting
RectSideRatio	0.05, 0.1, 0.15, 0.2
Capacity of a bucket	$2^6 \sim 2^9$ bytes
Size of a data object	1024 bytes
Size of coordinates	16 bytes
Size of a HC value	16 bytes
Size of a pointer	2 bytes

4.1. Evaluation of Access Latency

Fig. 11 shows the Access Latency performance when we fixed *RectSideRatio* to 0.1 and varied bucket capacity from 2^6 to 2^9 bytes. For both datasets, namely UNIFORM (Fig. 11 (a)) and REAL (Fig. 11(b)), DSPI is superior to the other two indexing schemes (HCI, DSI).

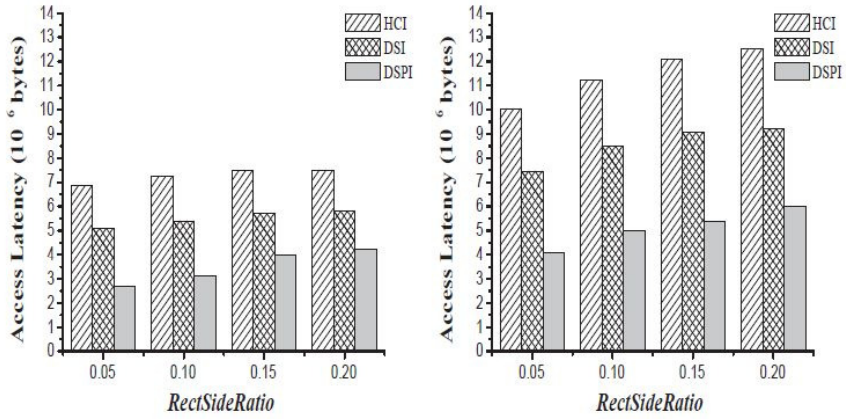
This is due to the fact that the overall length of the broadcast cycle in DSPI is shorter than those in the other two indices. Bucket capacity of UNIFORM dataset, DSPI decrease 51.7% and 30.6% in terms of the Access Latency over HCI and DSI. Bucket capacity of REAL dataset, DSPI decrease 51.1% and 16.6% in terms of the Access Latency over HCI and DSI. In other words, the index size of HCI and DSI is much larger than that of DSPI. In HCI, the whole index is interleaved m times with data objects, while in DSI, every frame has to maintain lots of information (e.g., HC values of $\log_r n_F$ frames and pointers to those frames) when a large number of data objects are concerned. Furthermore, in DSI, redundant information increases as the

number of data objects increases. Note that the Access Latency of DSPI, for the bucket size = 2^9, increases more rapidly than that of the others. This happens because the size of index segment, i.e., a number of contiguous index buckets, is much larger than the size of the directory (or leaf) cell. The Access Latency performance of DSPI tends to be affected by the bucket capacity due to its small index size.



(a)UNIFORM dataset (b) REAL dataset

Figure 11. Access Latency versus bucket capacity



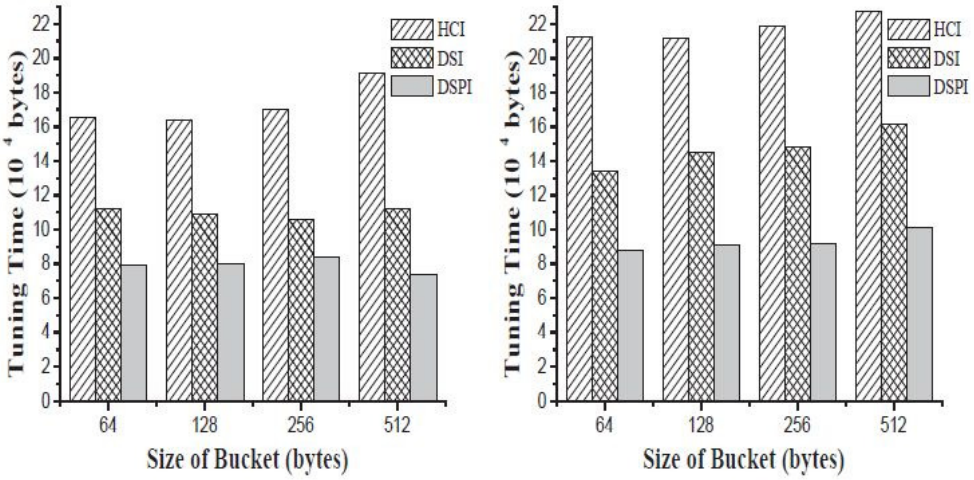
(a)UNIFORM dataset (b) REAL dataset

Figure 12. Access Latency versus *RectSideRatio*

For a more comprehensive evaluation, we fixed bucket capacity to 2^7 bytes and obtained the simulation results by varying *RectSideRatio*. Obviously, as depicted in Fig. 12, Access Latency of all the indices for both datasets gradually increases as the *RectSideRatio* increases. *RectSideRatio* of UNIFORM dataset, DSPI decrease 56.3% and 31.3% in terms of the Access Latency over HCI and DSI. *RectSideRatio* of REAL dataset, DSPI decrease 55.6% and 30.4% in terms of the Access Latency over HCI and DSI. However, DSPI shows better performance than the other indices.

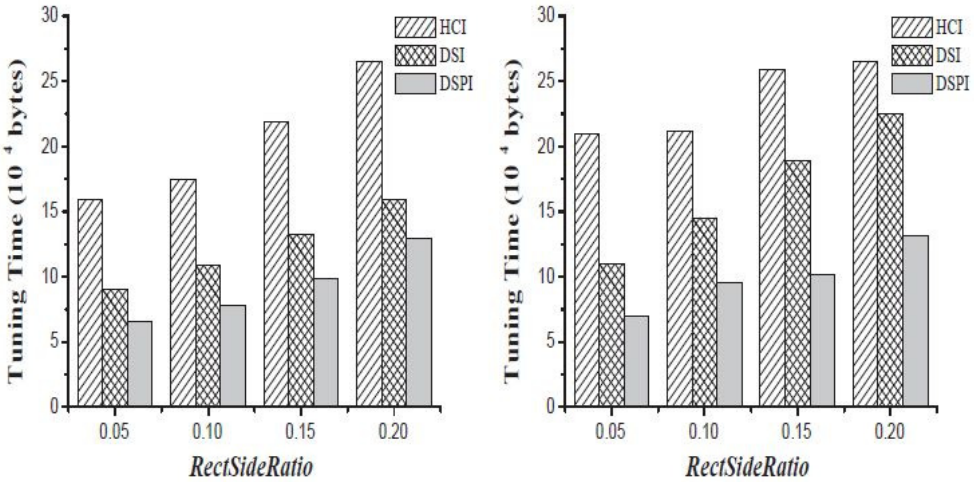
4.1. Evaluations of Tuning Time

Note that the Tuning Time affects the energy consumption on MCs. Here, we evaluated the Tuning Time performance. By fixing the *RectSideRatio* to 0.1 and varying bucket capacity from 2^6 to 2^9 bytes, Fig. 13 shows the Tuning Time performance of the three indexing schemes. Fig. 14 plots the Tuning Time performance under a fixed bucket capacity and varied *RectSideRatio*.



(a)UNIFORM dataset (b) REAL dataset

Figure 13. Tuning Time versus bucket capacity



(a)UNIFORM dataset (b) REAL dataset

Figure 14. Tuning Time versus *RecsideRatio*

As shown in both figures, DSPI outperforms HCI and DSI. Bucket capacity of UNIFORM dataset, DSPI decrease 55.3% and 17.1% in terms of the Tuning time over HCI and DSI. Bucket capacity of REAL dataset, DSPI decrease 57.4% and 24.1% in terms of the Tuning time over HCI and DSI. *RectSideRatio* of UNIFORM dataset, DSPI decrease 65.1% and 16.5% in terms of the Tuning time over HCI and DSI. *RectSideRatio* of REAL dataset, DSPI decrease 65.1% and 29.9% in terms of the Tuning time over HCI and DSI. This is because DSPI allows MCs to avoid reading

unnecessary data objects by offering the global view (MBRs) as well as the local view (coordinates of the data objects) of the underlying dataset. As mentioned in Section 2, both HCI and DSI are constructed based on the HC values of the underlying data objects instead of their exact coordinates. As a result, MCs have to listen to broadcast channel more than necessary. This incurs a huge amount of Tuning Time. Furthermore, as the order of Hilbert curve increases (due to the skewed distribution or a large number of data objects), spatial locality of the data objects in 2-dimensional space cannot be preserved in the 1-dimensional linear space. Therefore, MCs have to read much more data objects than necessary**.**

5. CONCLUSION

This paper addresses the problem of processing range queries on wireless broadcast streams. In order to support range queries efficiently, this paper proposed a novel index structure, called Distributed Space Partitioning Index, for processing range queries in wireless broadcast environments. DSPI utilizes the notion of hierarchical grids in order to allow MCs to selectively retrieve the required data in an efficient way in terms of Access Latency as well as Tuning Time. As demonstrated in the simulation study, DSPI outperforms the existing index schemes (e.g., HCI, DSI).

ACKNOWLEDGEMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2008578) && This research was funded by the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2013

REFERENCES

[1] T.Imielinski, S.Viswanathan and B.R.Bardrinath, Data on air: Organization and access, IEEE Transaction on Knowledge and Data Engineering (TKDE), 9(3): pp. 353- 372, 1997.

[2] A. Datta, D. VanderMeer, A. Celik and V. Kumar, Broadcast protocols to support efficient retrieval from databases by mobile users, ACM Transactions on Database Systems, 24(1): pp. 1-79, March 1999.

[3] C. Gostman and M. Linderbaum, On the metric properties of discrete space-filling curves, IEEE Transactions on Image Processing, 5(5): pp. 794-797, May 1996.

[4] J. Xu, B. Zheng, W.C. Lee and D.L. Lee, Energy efficient index for querying location-dependent data in mobile broadcast environments, Proc. 19th IEEE Conf. ICDE 03, 5(5): Bangalore, India, March 2003.

[5] B. Zheng, J. Xu, W.C. Lee and D.L. Lee, Energy conserving air indexes for nearest neighbor search, Proc. 9th Conf. EDBT 04, 5(5): Heraklion, Crete, Greece, March 2004.

[6] J. Xu, W.C. Lee, X. Tang, Q. Gao and S. Li, Exponential index: a parameterized distributed indexing scheme for data on air, Proc. 2nd ACM Conf. MobiSys 04, Boston, Massachusetts, USA: pp. 153-164, June 2004.

[7] B. Zheng, W.C. Lee and D. Lee, Spatial queries in wireless broadcast systems, Wireless Network, 10(6): pp. 723-736, December, 2004.

[8] W. Lee and B. Zheng, DSI: A Fully Distributed Spatial Index for Location-based Wireless Broadcast Services, Proc. 25th IEEE Conf. ICDCS 05, Columbus, Ohio, USA, pp. 349-358, June 2005.

[9] Spatial Datasets. available at http://www.rtreeportal.org.spatial.html.

[10] S. Ilarri, E. Mena, and A. Illarramendi, Location-Dependent Query Processing: Where We Are and Where We Are Heading, ACM Computing Surveys, 42 (3) (2010), pp. 1-73.

[11] B. Gedik, and L. Liu, Mobieyes: A Distributed Location Monitoring Service Using Moving Location Queries, IEEE Transactions on Mobile Computing, 5 (10) (2006), pp. 1384-1402.

[12] Y. D. Chung, An indexing scheme for energy-efficient processing of content-based retrieval queries on a wireless data stream, Information Sciences 177 (2) (2007), pp. 525-542.

[13] A. B. Waluyo, W. Rahayu, D. Taniar and B. Srinivasan, A novel structure and access mechanism for mobile data broadcast in digital ecosystems, IEEE Transactions on Industrial Electronics, 58 (6) (2011), pp. 2173-2182.
[14] J. Zhong, W. Wu, Y. Shi and X. Gao, Energy-efficient tree-based indexing schemes for information retrieval in wireless data broadcast, Lecture Notes in Computer Science 6588 (2011), pp. 335-351.
[15] Y. Kwon, H. Jung, Y. D. Chung, Monitoring continuous k-nearest neighbor queries in the hybrid wireless network, Journal of Zhejiang University SCIENCE, 12(3), 2011, pp. 213-220.
[16] H. Jung, B. K. Cho, Y. D. Chung and L. Liu, On processing location based top-k queries in the wireless broadcasting system, in Proceedings of ACM Symposium on Applied Computing, 2010, pp. 585-591.
[17] H. Jung, Y. D. Chung and L. Liu, Processing generalized k-nearest neighbor queries on a wireless broadcast stream, Information Sciences, 188, 2012, pp. 67-79.
[18] A. B. Waluyo, W. Rahayu, D. Taniar and B. Srinivasan, A novel structure and access mechanism for mobile data broadcast in digital ecosystems, IEEE Transactions on Industrial Electronics, 58 (6) (2011), pp. 2173-2182.
[19] K. Mouratidis, and S. Bakiras, Continuous Monitoring of Spatial Queries in Wireless Broadcast Environments, IEEE Transactions on Mobile Computing, 8 (10) (2009), pp. 1297-1311.

AUTHORS

Kwanho In received his B.S. degree in Computer Science from Dongguk University, Gyeongju, Korea, in 2009. He received the M.S. degree in information and communication engineering from Sungkyunkwan University, Suwon, Korea. His research interests include database system, spatial data management in mobile/pervasive environments.

HaRim Jung received his B.S. degree in Computer Science from Kwangwoon University, Seoul, Korea, in 2004. He received his M.S. and Ph.D. degrees in Computer Science and Engineering from Korea University, Seoul, Korea, in 2007 and 2012, respectively. Currently, he is a research fellow at the School of Information and Communication Engineering, Sungkyunkwan University, Suwon, Korea. His research interests include location-based services, spatial data management in mobile / pervasive environments and spatial big data management.

Hee Yong Youn received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1977 and 1979, respectively, and the Ph.D. degree in computer engineering from the University of Massachusetts at Amherst, in 1988. Currently he is a Professor of the School of Information and Communication Engineering, Sungkyunkwan University, Suwon, Korea, and the Director of the Ubiquitous Computing Technology Research Institute. His research interests include distributed and ubiquitous computing, system software and middleware, and RFID/USN.

Ung-Mo Kim received the B.E. degree in Mathematics from Sungkyunkwan University, Korea, in 1981 and the M.S. degree in Computer Science form Old Dominion University, U.S.A. in 1986. His Ph.D. degree was received in Computer Science from Northwestern University, U.S.A., in 1990. Currently he is a full professor of School of Information and Communication Engineering, Sungkyunkwan University, Korea. His research interests include data mining, database security, data warehousing, GIS and big data