Energy Efficient Data Access in Mobile P2P Networks

Kwangjin Park and Patrick Valduriez

Abstract—A fundamental problem for peer-to-peer (P2P) applications in mobile-pervasive computing environment is to efficiently identify the node that stores particular data items and download them while preserving battery power. In this paper, we propose a P2P Minimum Boundary Rectangle (PMBR, for short) which is a new spatial index specifically designed for mobile P2P environments. A node that contains desirable data item(s) can be easily identified by reading the PMBR index. Then, we propose a selective tuning algorithm, called Distributed exponential Sequence Scheme (DSS, for short), that provides clients with the ability of selective tuning of data items, thus preserving the scarce power resource. The proposed algorithm is simple but efficient in supporting linear transmission of spatial data and processing of location-aware queries. The results from theoretical analysis and experiments show that the proposed algorithm with the PMBR index is scalable and energy efficient in both range queries and nearest neighbor queries.

Index Terms—Moving objects, mobile computing, wireless data broadcasting, peer-to-peer.

1 Introduction

With the increasing popularity of portable wireless computers, mechanisms which allow information to be efficiently shared through mobile peer-to-peer (P2P) networks are of significant interest [1]. A mobile P2P environment is a set of moving nodes that communicate via short range wireless technologies such as IEEE 802.11, Bluetooth, or Wi-Fi [2], [3]. In particular, mobile nodes can be equipped with wireless P2P capabilities and this enables nodes to become parts of self-organizing and self-controlling with low-cost and easily deployed communication equipment [3], [4], [5], [6]. However, when the mobile environment consists of light-weight devices, such as Personal Digital Assistants (PDAs) and intelligent phones, the loss of network connectivity and scarce resources, e.g. low battery power, slow processors and limited memory, become primary issues of concern in order to efficiently support portable wireless devices [7].

Advances in wireless networks and positioning systems have led to the development of a new generation of mobile services which are called location-aware services. Location-aware services produce answers to queries, which depend not only on the data values, but also on the location where the query was issued [8], [9], [10]. With these mechanisms, a mobile node receives desirable information from its neighbors. The important classes of problems in location-aware services are *range* and *nearest-neighbor* search. Range search retrieves object(s) located inside of a query range, whereas nearest neighbor search retrieves the object that is the closest to a query point q. Examples of location-aware services in mobile P2P environments are: (i) a student in the university wants to share restaurant information with neighboring students, such as "the names and addresses of the French restaurants near the university" and (ii) a tourist wants to share the location of interest (i.e. souvenir shops, hotels or scenic spots) in the place being visited, such as "the names and addresses of the hotels near the train station". Then, the mobile client exchanges any desirable information from other nodes (clients) via short range wireless channels.

The client who issues the query can obtain desired data items with two different approaches, *periodic broadcast* and *on-demand*. In the on-demand approach, the client individually sends a request message to the server and obtains query results via a point-to-point channel. For example, when a tourist issues a query, such as "give me the names and addresses of the near hotels", a client individually sends a request message to the neighboring client and obtains a desirable result by establishing connection between the client and server (neighboring client). Then, the server processes query that the client submit on demand. In the periodic broadcast approach, the client listens to the broadcast channel and obtains query results via a broadcast channel. For example, a tourist wants to obtain the location of interest (e.g., near hotels) in the place being visited, such as "the names

• *P. Kwangjin is with the School of Electrical Electronics and Information Engineering, Wonkwang University, Iksan-Shi, Chunrabuk-do 570-749, Republic of Korea*
E-mail: kjpark@wku.ac.kr
• *P. Valduriez is with Atlas project team INRIA Sophia Antipolis Mediterranee LIRMM 161 rue Ada 34392 Montpellier Cedex 5 France*

and addresses of the near hotels". Then, the mobile client receives any desirable information from other clients via short range wireless broadcast channels. Compared to on-demand, periodic broadcast has the advantage of scaling up to serve an arbitrary number of clients without incurring additional cost at the server site. Moreover, the client does not need to send the details of its physical location to the server [11]. In wireless mobile environments, broadcasting the most frequently accessed data items saves bandwidth, as it cuts down the number of separate but identical responses to requests. A wireless data broadcast can be viewed as "storage on the air" and saves power on the client side by avoiding power consuming uplink transmissions [7], [12], [13]. To conserve scarce power in mobile clients, the broadcasting of data together with an index is effective [11], [12], [13]. Indexing (i.e. a directory of the file) can be used to guide the client in the listening process. The client is able to identify the contents of data items delivered from other nodes and to predict the arrival time of desired data by first accessing the broadcast index. Using an index gives the client the following advantages: (i) Avoid useless efforts since the client only sends requesting messages to the node with the desired data items by accessing *air indexing*[1]. Avoiding useless efforts helps to decrease communication costs among the nodes. (ii) Reduce the amount of time spent listening on the broadcast channel. The basic idea is to organize broadcast data such that the CPU can operate in low power sleep mode most of the time and can wake up to listen to the channel only when data of interest is broadcast.

Over the past few years, some studies have been introduced for broadcast-based mobile P2P approaches. However, most of these works only address the dissemination of reports about resources via wireless broadcast without taking into account the limited resources of the mobile client. Moreover, none of them considered location-based data dissemination and indexing for mobile P2P spatial queries in periodic broadcast systems. In this paper, we address the issues of supporting spatial queries of location-aware services via wireless data broadcast in mobile P2P networks. The main objectives of our work can be summarized as follows: (i) Present a new research direction of organizing spatial information and supporting spatial queries for resource-constrained mobile P2P networks. (ii) Introduce a new index method for broadcast-based mobile P2P environments. (iii) Develop a new index search algorithms.

We extend the previous work in the following aspects:

- We propose an efficient access index structure, called PMBR, for P2P spatial queries in wireless data broadcast. We describe how to construct a PMBR and how to deliver this index structure in mobile P2P environments.
- We develop a range and nearest-neighbor search algorithm based on the PMBR index to support P2P spatial queries.
- We extend ESS [23], [24] to efficiently support P2P spatial queries in wireless data broadcast systems. ESS only fits traditional client-server broadcast environments since the server broadcasts all data items of the whole universe.

We assume the existence of a set of static objects, such as hotels, gas stations and hospitals. Then, users share their information via short range wireless channels. Our goal is to conserve mobile clients' battery power and to minimize communication costs, i.e. number of messages among the mobile nodes in mobile P2P environments. To summarize, we make the following contributions:

- We propose PMBR, which is a rectilinear shape that completely contains the bounded object(s) of each node. PMBR provides the client with the ability to selectively contact or tune to other nodes, helping to reduce communication overhead among the nodes and to obtain fast answers. PMBR also facilitates energy conservation at the client by avoiding (or reducing) power consuming uplink transmissions.
- We propose a new selective tuning algorithm, called the Distributed exponential Sequence Scheme (DSS), which is designed for P2P spatial queries in a wireless data broadcast. The proposed algorithm provides clients with the ability to selectively tune for data items, preserving scarce power. Moreover, it is able to deal with the continuous movement of large numbers of mobile users and real-time requirements of spatio-temporal queries.
- We develop a cost model and conduct simulation experiments to validate the performance of our algorithm. The experimental results show that PMBR helps reduce the communication overhead among the nodes and that PMBR with DSS helps decrease battery power consumption.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 discusses spatial information transmission methods and Section 4 introduces our mobile P2P network environment. Section 5 describes the spatial index[2] and data dissemination and Section 6 describes our selective tuning algorithm. Section 7 gives the performance evaluation. Finally, Section 8 concludes.

1. Clients who have no prior knowledge of the contents of the broadcast data will access the directory from air [13].

2. A spatial index is a special access method used to retrieve spatial data from within the data-store.

2 RELATED WORK

In this section, we discuss: (i) P2P spatial data access in distributed spatial databases, (ii) indexing techniques for wireless data broadcast in mobile computing environments, and (iii) spatial queries on sequential-access broadcast channels.

2.1 P2P Spatial Data Access

Location-aware services in a P2P network have the advantage that peer nodes only maintain local information and there is no single point of failure. In [14], [15], the authors propose a technique to preserve spatial locality information while keeping some of the load balancing properties of distributed hash tables (DHTs) based systems. The proposed techniques use a hash function to preserve both spatial locality of data objects and constrained load balance in P2P systems. In [16], the authors propose an opportunistic approach, where a node propagates spatio-temporal resource information and obtains new reports in exchange. A moving node constantly receives availability reports from the peers it encounters. Since the number of reports saved and communicated by a peer may continuously increase, the authors employ a relevance function that prioritizes the availability reports in order to limit the volume of data exchange. In [17], authors propose the P2PR-tree (P2P R-tree), which is a spatial index specifically designed for P2P systems. The proposed scheme combines static space decomposition at the top index levels with dynamic space grouping at the lower levels of indexing to cope with highly skewed data distribution over peers. In [18], authors propose a sharing-based nearest neighbor query model, called MAPLE. MAPLE is designed for sharing of query results that are cached in the local storage of mobile clients. In [19], authors investigate the problem of efficient execution of spatial queries in sensor networks. Authors propose a peer-to-peer indexing structure, called peer-tree. Then, they present a peer-to-peer query processing model over peer-tree. The proposed model provides minimal energy consumption and response time with a distributed index structure. In [20], authors study multi-dimensional range queries in sensor networks. Authors introduce a new storage load balancing algorithm to achieve better balancing and reducing latency and the number of required operations. In [21], the authors propose EZCab, a real-life ubiquitous computing application for booking cabs in cities. EZCab discovers free cabs using a P2P short range wireless channel.

Even though the importance of spatial data access in mobile computing environments has been established for traditional queries, such as range and nearest neighbor queries, there is limited work on energy efficient query processing in mobile P2P environments. Most of the previous studies for P2P spatial data access only discussed in on-demand approach. An on-demand approach may increase message cost between a client node and a service provider node. This may increase a mobile client's energy consumption and search time. To efficiently support P2P spatial queries, contact to unqualified service provider should be avoided since useless contact may increase communication costs. Therefore, to facilitate energy conservation, a spatial index can be used to support selective tuning (or contacting) for spatial queries in mobile P2P networks.

2.2 Indexing Techniques

In traditional client-server broadcast environments, reducing both *tuning time*[3] and *access time*[4] is the most important issues in terms of power conservation and correct (or prompt) answer. The (1, m) index [13] is a well known air index techniques. In this method, the index is broadcast m times during a single broadcast cycle. The broadcast index is broadcast every fraction $\frac{1}{m}$ of the broadcast cycle. Selective tuning is accomplished by multiplexing an index with the data items in the broadcast. In general, the fastest access time in a broadcast cycle is obtained when there is no index, since the size of the entire broadcast cycle is minimized but this increases the tuning time. In this case, the average latency time is $\frac{N}{2} + d$, where N denotes the number of data objects and d denotes the download time for data objects. On the other hand, increasing the number of index segments in a single broadcast cycle reduces the average *probe wait*[5] time, but increases the access time because of the additional index information. In this case, the probe wait time for the next index is equal to $\frac{1}{2} \times (i + \frac{N}{m})$ and the wait time for desired data object is equal to $\frac{1}{2} \times (N + i \times m)$, where i denotes the size of the index. Since the access time is the sum of the probe wait time for the index and the wait time for the desired data object, the average access time for the (1, m) index is equal to $\frac{1}{2} \times ((m+1) \times i + \frac{1}{m} + 1 \times N) + d$ (see [12] for the details). The (1, m) index has a major drawback: the probe wait time may increase the average access time. Thus, data broadcast should be considered for sequential-access broadcast ordering in order to support the linear streaming property of wireless data broadcast channels.

3. *Tuning time* represents the amount of time spent by a client listening to the channel.

4. *Access time* represents the period of time elapsed from the moment a user issues a query to the client to the moment when the required data item is received by the client.

5. The average duration for getting to the next index segment is called the probe wait [12].

2.3 Spatial Queries on Sequential-access Broadcast Channels

Existing index structure and search algorithms are designed for traditional spatial databases and do not consider the time-series characteristics of the air index [10], [11], [24], [30]. Therefore, they cannot be easily deployed in wireless data broadcast systems. For example, most of the existing studies on spatial search are based on indexes that store the locations of the indexed objects, e.g. the well-known R-tree[6] and its variants such as R^+-tree or R^*-tree. However, R-tree-based methods are better supported by random access storage, such as Random Access Memory (RAM) and disk, but not by wireless data channels. A search algorithm based on the R-tree typically expands the search space around the query point using a branch-and-bound approach. This usually requires backtracking before the target leaf node is found. Information is broadcast based on a pre-defined sequence and it is only available at the moment when it is broadcast. Backtracking tree search causes a serious problem for sequential access media (e.g. wireless data broadcast channel) [10].

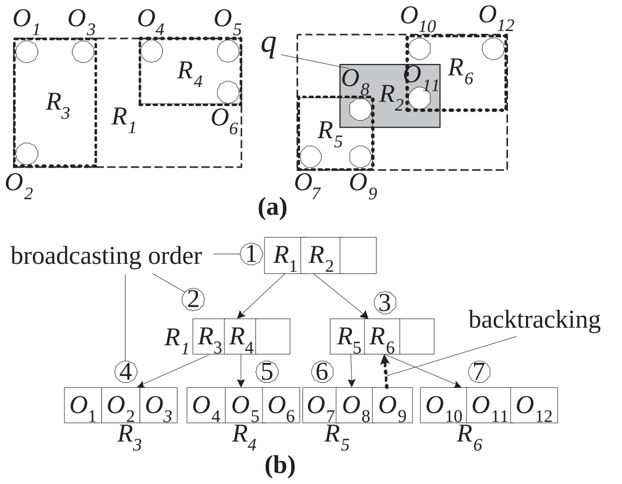


Fig. 1. R-tree Index for the Running Example. (a) Minimal bounding rectangles and query region q. (b) R-tree index.

Let us consider Figure 1 as an example. Suppose that the client wants to identify the objects inside q (see Figure 1(a)). As shown in the figure, there are 12 objects and objects that are close in space (e.g. O_1, O_2, O_3) are clustered as a minimal bounding rectangle (MBR). Let us assume that the index segments are broadcast in hierarchical order, i.e. $<R_1$, R_2 R_3, R_4, R_5, R_6, O_1, O_2, $O_{12}>$ (see Figure 1(b)). Now, the R-tree answers a window query q as follows: (i) The root (R_2) is first retrieved, (ii) R_5 and the entries inside R_5 (i.e. O_7, O_8, O_9) are checked with q since *int* (q, R_5), (iii) it goes back and checks R_6 since *int* (q, R_6), while R_6 has already been broadcast in the current cycle. Now, the node must wait for the next broadcast cycle and this significantly extends the access latency. Therefore, a promising direction for spatial indexing with a wireless data broadcast system is to extend the general framework in order to support the *linear streaming property*[7] of wireless data broadcasts.

In a previous paper [23], [24], [25], we proposed a novel broadcast-based spatial data dissemination and selective tuning algorithm that provide the clients with the ability to perform selective tuning and assist in reducing the client's tuning time. The basic idea is to use exponential pointers from each data item. With the exponential pointer, each data object contains pointers that contain the IDs, locations, and arrival times of the data items that will subsequently be broadcast. Each client utilizes an exponential pointer from each data item for the purpose of reducing energy consumption. In [26], we proposed a algorithm to support dynamic, continuous nearest neighbor queries in wireless broadcast environments. To enable clients to find the exact answer for moving queries, we defined the GR, which divides a query line segment into disjoint lines where the nearest neighbor of any point inside a GR is the same.

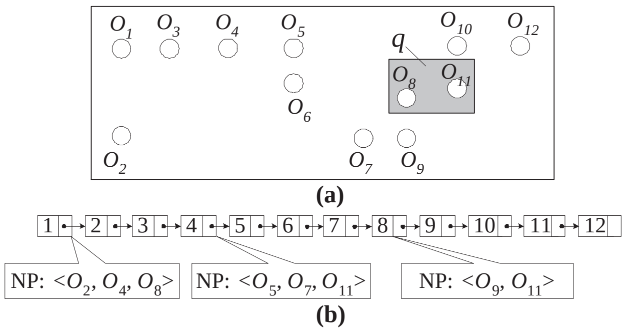


Fig. 2. ESS for a Range Query. (a) Data objects and query region q. (b) Next Pointer.

Let us suppose that the client wants to find the objects inside q (see Figure 2(a)). Let the broadcasting sequence be in horizontal order, i.e. from the leftmost data object to the rightmost data object. The client begins to tune into the broadcast channel from the leftmost one (i.e. O_1). As shown in Figure 2(b), O_1 has the following information for NP: O_2, O_4 and O_8. After the client receives O_1, it switches to power save mode until O_4 appears on the broadcast channel, since (x-coordinate of O_4) < (x-coordinate of q). The client wakes up when O_4 appears on the broadcast channel and obtains NP of O_5, O_7 and O_{11} from

6. The R-tree is a classical spatial index structure. The basic idea is to approximate a spatial object with a minimal bounding rectangle (MBR) and to index the MBRs recursively [1], [22].

7. The server disseminates data items via one-dimensional wireless broadcast channel and the client sequentially accesses them.

O_4. Then, the client switches to power save mode again until O_7 appears, since (x-coordinate of O_7) $<$ (x-coordinate of q). The client repeatedly switches between power-save and wake up modes until the desired data object is retrieved. Notice that if the client wakes up before O_n arrives, it can be guaranteed that it does not miss any objects inside the query range q, where (x-coordinate of O_n) $\geq$ (x-coordinate of q). In [11], the authors present a supporting Nearest Neighbor (NN) search in wireless data broadcast services. Then, they address the search algorithms for NN in wireless data broadcast, which requires revision in order to fit the linear streaming property. Two air indexing techniques are proposed, namely R-tree air index and Hilbert-Curve-based air index, for processing the NN search on air. In [27], the authors present techniques for scheduling a spatial index tree for broadcast in a single and double channel environment. The algorithms executed by the clients aim at minimizing latency and tuning time.

ESS and tree-based spatial index techniques only fit traditional client-server broadcast environments since the server broadcasts all data items of the whole universe. Moreover, most of the previous studies for P2P spatial queries focus only on point-to-point communication. Therefore, to efficiently support P2P spatial queries, sharing and dissemination of data items should be distributed.

3 Spatial Information Transmission for Mobile P2P Environments

In this section, we discuss the different methods of spatial information transmission, then we compare how these methods consume battery power. In particular, we focus on energy consumption which is a main issue in mobile P2P environments.

A mobile P2P environment is a set of moving nodes equipped with short range wireless technologies. Each node is aware of its position through a Global Positioning System (GPS) device. In the mobile P2P environment, most mobile devices are restricted in their power source and communicate on limited bandwidth wireless channels. Hence, saving power is critical [12].

In general, there are two different approaches to obtain desired information in mobile P2P environments, *periodic broadcast* and *on-demand*. Periodic broadcast has the advantage of scaling up to serve an arbitrary number of nodes without incurring additional costs at the transmission site. Moreover, a client does not need to individually submit a request message to others in order to obtain the desired information. However, the client must continuously stay in active mode and tune into the broadcast channel until it obtains desired information, although it may not be the desirable information. Thus, the client may waste a lot of energy. With on-demand, a client must contact other nodes, although they may not have the desired information since the client is unable to determine whether a neighbor node has the desired information or not before contacting it. In order to support efficient query processing, it is important to efficiently transmit information to clients in mobile P2P environments.

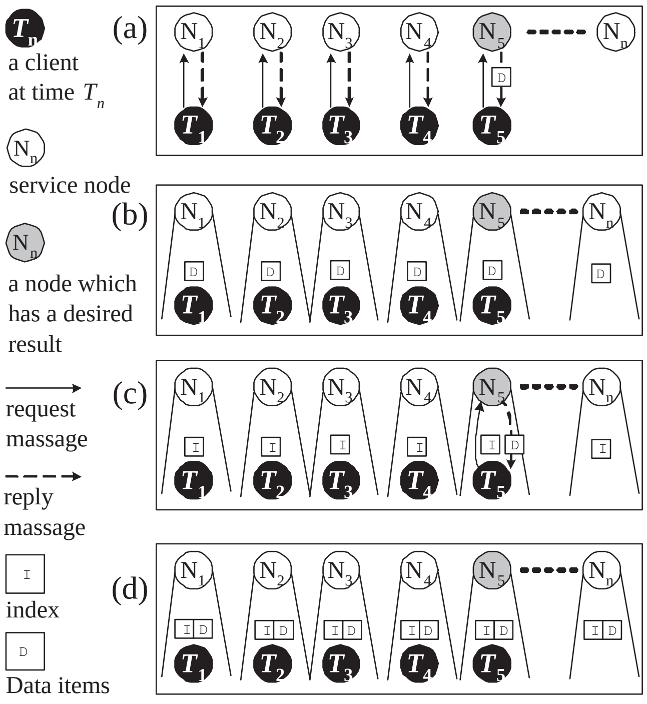


Fig. 3. Data Access. (a) *on-demand*. (b) *periodic data*. (c) *periodic index*. (d) *periodic index with selective data*.

We now discuss the four main methods based on periodic broadcast and on-demand approaches. (i) *Point-to-point*: the client obtains a desirable result by point-to-point communication (Figure 3(a)), (ii) *Periodic data*: the client obtains a desirable result by tuning to the periodic data broadcast (Figure 3(b)), (iii) *Periodic index*: the client obtains a desirable result by selective contacting of the nodes (i.e. using index) and sends a request message only to the node(s) who has the desired information (Figure 3(c)), and (iv) *Periodic index with selective data*: the client obtains a desirable result by selective contacting of the nodes and receives the requested data items with a selective tuning algorithm, such as ESS or R-tree spatial indexing (Figure 3(d)).

Now, let us compare the average energy consumption of the four different methods discussed above with the following notations. Let ε_{Req} and ε_{Ack} be the energy consumption for sending a query message and receiving an acknowledge message from node i, respectively. Let ε_{data}, ε_{data_sel}, and ε_{index} be the energy consumption for receiving data items during a whole broadcast cycle, the energy consumption for receiving data items with selective tuning, and the energy consumption for receiving index messages from node i, respectively. Let ε_d and n be the energy consumption for the sum of downloading all required data item(s) and the number of node contacts until the client finds the final result, respectively.

(i) *Point-to-point*: We first consider the average energy consumption (AEC, for short) of *Point-to-point*. In this method, the client submits request massages and receives acknowledge messages from n nodes (e.g. from N_1 to N_5 in Figure 3(a)). Then, the client obtains the final result from the n^{th} node (e.g. N_5 in Figure 3(a)). Thus, AEC of *Point-to-point* is:

$$n \times (\varepsilon_{Req} + \varepsilon_{Ack}) + \varepsilon_d \tag{1}$$

In *Point-to-point*, the value of n affects the energy consumption significantly since the client must repeatedly send a query and receive an acknowledgement message. Furthermore, the uplink transmission (to submit a request massage) consumes much more battery power than the downlink transmission. Therefore, it causes a dramatic increase in energy consumption as the number of contact nodes increases.

(ii) *Periodic data*: Let us consider the AEC of *Periodic data*. In this method, the client tunes the broadcast data items from the n nodes (e.g. from N_1 to N_5 in Figure 3(b)). Then, the client obtains the final result from the n^{th} node (e.g. N_5 in Figure 3(b)). Thus, the AEC of *Periodic data* is:

$$n \times \varepsilon_{data} + \varepsilon_d \tag{2}$$

In *Periodic data*, the size and the number of data items affect the energy consumption significantly since the client must stay in active mode until it receives the desired data. This causes a high level of energy consumption and latency as the number of contact nodes increase.

(iii) *Periodic index*: Let us consider the AEC of *Periodic index*. In this method, the client tunes indexes from the n nodes (e.g. from N_1 to N_5 in Figure 3(c)). Then, the client submits a request message and obtains the final result from the n^{th} node (e.g. N_5 in Figure 3(c)). Thus, the AEC of *Periodic data* is:

$$n \times \varepsilon_{index} + \varepsilon_{Req} + \varepsilon_d \tag{3}$$

In the *Periodic index*, the size and the number of data items do not heavily affect the energy consumption since the client sends a request message only to the node with the desired information.

(iv) *Periodic index with selective data*: Let us consider the AEC of the *Periodic index with selective data*. In this method, the client tunes indexes from the n nodes (From N_1 to N_5 in Figure 3(d)). Then, the client selectively tunes and obtains the final result from the n^{th} node (N_5 in Figure 3(d)). Thus, the AEC of *Periodic data with selective data* is:

$$n \times \varepsilon_{index} + \varepsilon_{data\_sel} + \varepsilon_d \tag{4}$$

In *Periodic index with selective data*, since the client can also selectively tune the data items without sending a request message to other nodes, it is efficient in terms of power and resource consumption, ***especially when many users request the same data at the same time***.

In summary, the energy consumption of *Point-to-point* and *Periodic data* may significantly increase as the size and the number of data items or the value of n increase. On the other hand, *Periodic index* and *Periodic index with selective data* do not heavily depend on the size, the number of data items, and the value of n. These results lead us to the conclusion that *Periodic index* and *Periodic index with selective data* are suitable for resource-constrained mobile P2P networks.

4 Mobile P2P Network Environment

In this paper, we consider a mobile P2P network with a geometric location model, where locations are represented in terms of 2D coordinates. Each mobile node has a limited transmission range that is capable of communicating with the neighboring clients within a maximum of a few hundred meters. One example is an 802.11 hotspot or a PDA with Bluetooth support [28]. In this paper, we do not assume any infrastructure, such as a server, centralized databases, or a base station. There is neither a centralized directory nor any precise control over the network topology or file placement. Instead, we focus on providing location-based information services in a decentralized manner that enables more powerful accesses to data in P2P networks. To find a file, a client queries its neighbors. The query is propagated to all neighbors within a certain radius. Two mobile clients can communicate with each other directly, if they are in the transmission range of one another. The client moves from one location to another and communicates with other clients in high density places, such as a train station, university, or central park. Each node is allowed to tune a single broadcast channel at a time and to maintain storage to share spatial information. We assume that the size for the data items is the same and the client broadcasts data items in horizontal order based on their locations. The client's mobility pattern follows the *Random Waypoint Mobility Model* [31], which is widely used.

After a mobile client receives a query from its user, it tunes into the wireless data broadcast channel and receives data in accordance with the specified query conditions. The client can collect cached spatial data from other nodes to harvest existing results to complete its own spatial query. In this paper, we assume two different types of clients based on the source of electric power as follows:

- Adequate Resource Device (ARD): the client does not consider battery power consumption limitations, since it can be supported by electric power supply. Examples include an information center, hotel, restaurant or a mobile user with a vehicle. In this case, the client periodically broadcasts information to the interested group or potential customer. An example of this application includes location-aware advertisements, such as “send e-coupons to all potential users that are within 2km of our hotel”. The service provider acts as a server and periodically broadcasts data items with PMBR index (a spatial index represented by a bounding box which is a rectilinear shape that completely contains the bounded object(s)), via a wireless data broadcast channel.
- Limited Resource Device (LRD): the client has limited battery power so it is necessary to minimize battery power consumption. In this case, the client periodically broadcasts PMBR or tunes only.

The client who issues a query obtains desired data items from two different methods:

- *Periodic index*: A mobile client submits a request, which consists of a query (what it wants) and the query’s issuing location (where it is located) based on the value of PMBR from the LRD client, to the client. Then, the LRD client returns the result to the client.
- *Periodic index with selective data*: A mobile client obtains desirable information from the broadcast channel without revealing him or her to other clients.

In summary, each node acts as a server or a client while moving or staying in mobile P2P environments. Moreover, a node chooses their roles such as Adequate Resource Device (ARD) and Limited Resource Device (LRD) based on the source of electric power in a local area.

The above assumptions combine the advantages of existing information transmission methods to provide energy efficient query processing in mobile P2P environments.

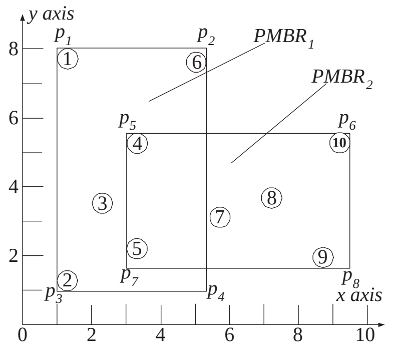


Fig. 4. PMBR Index

5 P2P Minimum Boundary Rectangle (PMBR) index and Data Dissemination

In this section, we propose a new spatial indexing method, called PMBR, specifically designed for wireless data broadcasting in mobile P2P environments. Then, we introduce the broadcast sequence ordering and query processing with a PMBR index algorithm.

The client defines the P2P minimal boundary rectangle, called PMBR, which is a rectilinear shape that completely contains the bounded object(s) of each node. The base index structure for the proposed PMBR index is similar to the R-tree. The PMBR index tree is constructed based on a Minimal Bounding Rectangle (MBR). The PMBR consists of four points, such as P_1, P_2, P_3, and P_4, and the locations of objects, such as O_1, O_2, O_3, O_4, O_5, and O_6 inside the $PMBR_1$ in Figure 4.

```
Parameters:
$S$: data objects located inside of PMBR
$O_l$: a data object that is located at the leftmost extremity in the PMBR
$O_c$: a data object that is currently located at the leftmost extremity in
the PMBR
$O_{c+1}$: a data object that is currently located at the leftmost extremity in
the PMBR with the exception of $O_c$
$x_c$, $x_{c+1}$: $x$-coordinate of $O_c$ and
$x$-coordinate of $O_{c+1}$ ,respectively
x-dist, y-dist: the counter //initial value=0
Input: IDs and locations of $S$;
Output: selection result for broadcast sequence;
Procedure:
1: sort objects of $S$ by ascending order according to the $x$-dimension ordering
2: while ($S \neq \emptyset$)
3:    for each object $O \in S$
4:        find $O_l \in S$; $O_c = O_l$; exclude $O_l$ from $S$ // if more than two objects
have the same $x$-axis value, select upper object first
5:        for each object $O \in S$
6:            find $O_l \in S$; $O_c + 1 = O_l$; exclude $O_l$ from $S$ // if more than
two objects have the same $x$-axis value, select upper object first
7:                if (value of $|x_c - x_{c+1}|$) > (value of $|y_c - y_{c+1}|$)
8:                    then increase x-dist
9:                else increase y-dist
10: end while
11: if x-dist > y-dist
12:    then select HB for the broadcast data object // if more than two objects
have the same $x$-axis value, broadcast upper object first
13: else select VB for the broadcast data object // if more than two objects
have the same $y$-axis value, broadcast leftmost object first
```

Fig. 5. Data Dissemination Algorithm: broadcasting order

Let us consider Figure 4 as an example. The ARD client which contains information for $PMBR_1$ broadcasts the following information: (i) PMBR: P_1 (x:1, y:8), P_2 (x:5.3, y:8), P_3 (x:1, y:1), P_4 (x:5.3, y:1), (ii) Broadcast order: vertical (i.e. from top to bottom), (iii)

```
Parameters:
PMBR$_{ARD}$: PMBR from ARD client
PMBR$_r$: final result (or set) of $q$ //initially PMBR$_r$=$\emptyset$
PMBR$_p$: partial result (or set) of $q$ //initially PMBR$_p$=$\emptyset$
Input: PMBRs and objects inside of each PMBR;
Output: PMBR$_r$;
Procedure:
1: while (obtain PMBR$_r$)
2:   tune and check PMBR
3:   if PMBR $\supseteq q$
4:       if PMBR$_{ARD}$ then
5:           selective tune until obtain PMBR$_r$
6:       else request PMBR$_r$ to LRD client
7:   else if PMBR $\subset q$
8:       if PMBR$_{ARD}$ then
9:           selective tuning until obtain PMBR$_p$
10:          PMBR$_r$=PMBR$_r$$\cup$PMBR$_p$
11:      else request PMBR$_p$ to LRD client
12:          PMBR$_r$=PMBR$_r$$\cup$PMBR$_p$
13:  else tune another broadcast channel
14: end while
15: return PMBR$_r$
```

Fig. 6. Query Processing with PMBR Index Algorithm

Data IDs and locations: O_1 (x:1.2, y:7.9), O_6 (x:5.2, y:7.8), O_4 (x:3.4, y:5.2), O_3 (x:2.5, y:3.6), O_5 (x:3.2, y:2.1), O_2 (x:1.2, y:1.2).

The LRD client only broadcasts PMBR, whereas the ARD client broadcasts the following information: (i) PMBR, (ii) broadcast ordering: horizontal or vertical, and (iii) IDs and 2D coordinates of the data objects via wireless data broadcast channel. A simple sequential broadcast can be generated by linearizing the 2D coordinates in two different ways. Linearizing can be completed by horizontal broadcasting (HB) or vertical broadcasting (VB). In HB, the ARD client broadcasts the LDD (location dependent data) in horizontal order, that is, from the leftmost coordinate to the rightmost coordinate. In VB, the ARD client broadcasts the LDD in vertical order, that is, from the top coordinate to the bottom coordinate. Figure 5 shows the pseudo-code for the ARD client to decide whether to broadcast the data using HB or VB. Then, Figure 6 shows the pseudo-code for query processing with PMBR. The client who issues a query selectively tunes a broadcast channel of node i and merges the partial results in order to obtain the final result. As described, the client does not need to tune the broadcast channel of node i if it does not have the desired data items since it already checks the PMBR via wireless data broadcast channel(s). Therefore, avoiding useless efforts helps to decrease message cost caused by the neighboring peers and to reduce the amount of time spent listening on the broadcast channel.

6 Selective Tuning Algorithm

In this section, we propose a selective tuning algorithm for range and NN queries in mobile P2P environments. In this algorithm, the ARD client divides PMBR based on the number (or distribution) of data objects for the purpose of selective tuning. Then, the client who tunes the ARD client's broadcast channel is able to predict the approximate arrival time of the desired data objects. To facilitate selective tuning via wireless data broadcast channels, each data object includes the following information:

- ID and sequence number of broadcasts;
- x-coordinates of the leftmost and rightmost located object;
- Partition Cells (PC_s) and the Number of objects inside its Cell (N_c);
- Arrival time of a data item;
- O_{first}: the first broadcast data object.

6.1 Range Query Processing

A range query returns the objects inside of q [7]. An example of a range query is: "show me all gas stations within 10 km of my location". In this section, we focus on energy efficient selective tuning for range query processing.

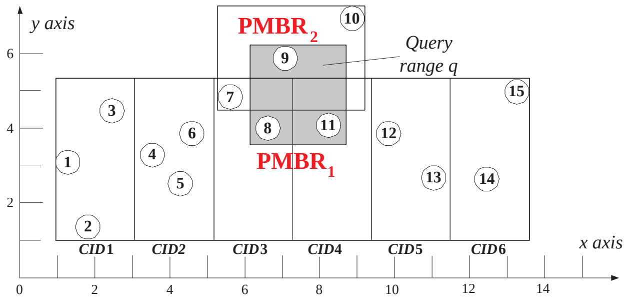


Fig. 7. Index-based Range Query

Let us consider the example in Figure 7. Assume that ARD clients of C_1 and C_2 broadcast PMBR$_1$ and PMBR$_2$ and objects inside each PMBR, respectively. Let the PC_s value of PMBR$_1$ be 6 and that of PMBR$_2$ be 1. The client who issues a query (C_3) wants to obtain objects from query range q (shaded area). The client C_3 tunes both PMBRs of C_1 and C_2 and does the following steps:

STEP (1): tune into the broadcast channel of C_1.

STEP (2): identify the distance between O_1 and O_{15} based on PMBR$_1$.

STEP (3): divide PMBR$_1$ into 6 partitions according to the value of PC_s.

STEP (4): find a cell which contains (or intersects) q, i.e. CID (cell ID) 3 and CID 4.

STEP (5): check the sum of the objects inside the cells located left-side of CID 3, i.e. the sum of N_c before CID 3=6.

STEP (6): switch to doze mode until O_7 arrives.

STEP (7): wake up when O_7 arrives and tune until it obtains PMBR$_{r<1>}$=$\{O_8, O_{11}\}$, where PMBR$_{r<1>}$ denotes result set of q from PMBR$_1$.

STEP (8): disconnect the broadcast channel of C_1.

STEP (9): tune into the broadcast channel of C_2 and

obtain PMBR$_{r<2>}$={O$_9$} (Notice that, in this case, it does not need to divide and check the sum of objects since the PC_s value of PMBR$_2$=1).
STEP (10): the final result of q={PMBR$_{r<1>}$ ∪ PMBR$_{r<2>}$}= {O$_8$, O$_9$, O$_{11}$}.

Figure 8 shows the pseudo-code for range query processing based on the above 10 steps.

```
Parameters:
S$_{MBR}$: set of currently tuned PMBR, where initially S$_{MBR}$=∅
C$_{left}$: a leftmost-cell which includes (or intersects) query region
C$_{right}$: a rightmost-cell which includes (or intersects) query region
S$_{left}$: sum of objects which located the left-side of C$_{left}$
S$^{th}$ object: object that broadcasts just before the object inside of C$_{left}$
Input: PMBRs and objects inside of each PMBR;
Output: PMBR$_r$;
Procedure:
1: read PMBR (STEP 1)
2: divide RMBR according to the value of $PC_s$ (STEP 3)
3: find C$_{left}$ and C$_{right}$
4: check S$_{left}$ and identify S$^{th}$ object (STEP 5)
5: while (obtain PMBR$_r$)
6:    for each object O$_i$
7:       if ($x$-coordinate of O$_i$) $\geq$ ($x$-coordinate of C$_{left}$)
8:          then switch to doze mode until O$_{first}$ arrives
9:       else
10:         while(S$^{th+1}$ object arrives)
11:            switch to doze mode (STEP 6)
12:         end while
13:         while (($x$-coordinate of O$_i$) $\leq$ ($x$-coordinate of C$_{right}$))
14:            if O$_i$ is located inside of $q$
15:               then PMBR$_r$ = PMBR$_r$ ∪ O$_i$ (STEP 7)
16:         end while
17: end while
18: return PMBR$_r$
```

Fig. 8. Range Query Processing Algorithm

Let us introduce two definitions used for selective tuning and contacting nodes for range queries.

Definition 1: (final decision for range query). Given a search range q, if cell(s) $\mathcal{C}$ of PMBR$_i$ or cell(s) $\mathcal{C}$ of S$_{MBR}$ contains q, the object(s) inside of the $\mathcal{C}$ is (are) possible result object(s) of q.

Definition 2: (selective tuning for range query). Given a search range q, although the client stays in sleep mode and misses the following cell(s) $\mathcal{C}$ from PMBR$_i$: (x-coordinate of $\mathcal{C}$) < (x-coordinate of C$_{left}$), (x-coordinate of $\mathcal{C}$) > (x-coordinate of C$_{right}$), or (x-coordinate of $\mathcal{C}$) > (x-coordinate of cell which contains q), (x-coordinate of $\mathcal{C}$) < (x-coordinate of cell which contains q), it can be guaranteed that it does not miss any object(s) inside of q from PMBR$_i$.

It is sufficient to recognize that the client can stop tuning PMBR$_i$ from ARD clients and return the PMBR$_r$ if cell(s) contains q, as described in Definition 1. If the client satisfies Definition 1, it does not need to tune the other PMBR and can return the object O$_i$ as the final result. Definition 2 introduces cells which do not need to be tuned during the range query processing. Thus, even though the client does not tune the objects inside of the cells, it can still be guaranteed that it does not miss any object(s) inside of q from PMBR.

6.2 Nearest Neighbor Query Processing

An important class of location-aware query is the nearest-neighbor (NN) search, which retrieves the object that is the closest to a query point [7], [29]. In this section, we focus on an energy efficient selective tuning algorithm for NN query processing. In order to reduce the tuning time, the client should know when it stops tuning PMBR$_i$ from ARD clients.

Let us introduce the following definition used for the final decision for NN queries.

Definition 3: (final decision for NN query). Given a search range q, if PMBR$_i$ or S$_{MBR}$ contains MBC, the object inside of MBC is NN.
If it satisfies Definition 3, the client does not need to tune another PMBR and returns NN as the final result. Let MBC be the Minimal Boundary Circle that contains a *possible object for the nearest neighbor* (NN$_p$) from PMBR.

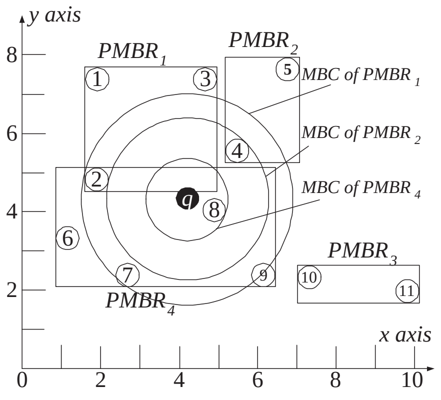


Fig. 9. Index-based NN Query

Let us consider the example in Figure 9. The client desires to find the NN from the query point q. Assume that ARD clients of C$_1$, C$_2$, C$_3$ and C$_4$ broadcast PMBR$_1$, PMBR$_2$, PMBR$_3$ and PMBR$_4$ and objects inside of PMBR$_i$, respectively. Assume that the client tunes PMBRs in the following order: <PMBR$_1$, PMBR$_2$, PMBR$_3$, PMBR$_4$>.
STEP (1): The client reads PMBR$_1$ and the objects inside PMBR$_1$ from C$_1$, and then draws a MBC of PMBR$_1$. Currently, NN$_p$=O$_2$.
STEP (2): The client checks the intersection of PMBR$_2$ with the MBC of PMBR$_1$. Since *int* (PMBR$_2$, MBC of PMBR$_1$), the client reads objects inside PMBR$_2$ from C$_2$. Currently, NN$_p$=O$_4$ and the client draws a MBC of PMBR$_2$.
STEP (3): The client checks the intersection of PMBR$_3$ with the MBC of PMBR$_2$. Since PMBR$_3$ neither contains nor intersects with the MBC, the client skips to tune and check the objects from PMBR$_3$. Currently, NN$_p$=O$_4$.
STEP (4): The client checks the intersection of PMBR$_4$ with the MBC of PMBR$_2$. Since *int* (PMBR$_4$, MBC of PMBR$_2$), the client reads PMBR$_4$ and objects inside

of PMBR$_4$ from C$_4$. Currently, NN$_p$=O$_8$ and S$_{MBR}$ of PMBR$_1$ and PMBR$_4$ contains MBC. Since the current condition satisfies Definition 3, O$_8$ is the final result of NN.

Figure 10 shows the pseudo-code for NN query processing based on the above 4 steps.

```
Input: PMBRs and objects inside of each PMBR;
Output: NN;
Procedure:
1: read PMBR and objects inside of PMBR (STEP 1)
2:   find NN$_p$ and draw MBC
3: while (identify NN)
4: for each PMBR$_i$
5:   read PMBR$_i$
6:   if int (PMBR, MBC) OR MBC contains PMBR (STEP 2)
7:       then selective tune objects inside of PMBR$_i$ // see Figure 11 for selective tuning algorithm
8:       find NN$_p$ and draw MBC // if NN$_p$ does not changed, MBC will not be changed (STEP 3)
9:           if PMBR$_i$ contains MBC then NN$_p$=NN
10:          else if S$_{MBR}$ contains MBC then NN$_p$=NN (STEP 4)
11:          else S$_{MBR}$ $\cup$ PMBR$_i$ and NN$_p$=NN$_p$
12:  else NN$_p$=NN$_p$ // skip to tune the objects inside of PMBR
13: end while
14: return NN
```

Fig. 10. NN Query Processing Algorithm

We now describe the steps taken by the client to process the "exact result" of a selective NN search: Let us define the following notations: (i) C$_q$ be a cell which includes a query point, (ii) *safety_circle* be the minimum circle centered at q which contains C$_q$ or the nearest cell which contains at least one object from the left-side of C$_q$ (see Figure 12), (iii) C$_{safe}$ be the leftmost cell $\mathcal{C}$, where *int* (*safety_circle*, $\mathcal{C}$), e.g. CID_3 in Figure 12, (iv) S$_{safe}$ be the sum of objects which located the left-side of C$_{safe}$, and (v) S$_{safe}{}^{th}$ object be the object that is broadcast just before the object inside of C$_{safe}$.

STEP (1): divide PMBR according to the value of PC_s and identify C$_q$. Then, if C$_q$ contains at least one object, go to STEP 2 else go to STEP 3.

STEP (2): draw *safety_circle* centered at q which contains C$_q$ and go to STEP 4.

STEP (3): draw *safety_circle* centered at q which contains at least one object from the left-side of C$_q$ and go to STEP 4.

STEP (4): if (x-coordinate of O$_i$)>(x-coordinate of C$_{safe}$), then go to STEP 5, else go to STEP 6.

STEP (5): switch to doze mode until S$_{safe}{}^{th+1}$ object arrives from the next broadcast cycle. Go to STEP 7.

STEP (6): switch to doze mode until S$_{safe}{}^{th+1}$ object arrives from the current broadcast cycle.

STEP (7): compare the distance between the current broadcast object (O$_i$) and q with NN$_p$ and q until (*dist* ((x-coordinate of O$_i$),(x-coordinate of q)) $\geq$ *dist* (NN$_p$, q)) (see shaded fan shape in Figure 12). If *dist* (O$_i$, q) $<$ *dist* (NN$_p$, q), then O$_i$=NN$_p$.

STEP (8): return the final result of NN$_p$ as NN.

Figure 11 shows the pseudo-code for NN query processing based on the above 8 steps.

```
Parameters:
C$_{near}$: the nearest cell which includes at least one object from the left-side of C$_p$
Input: PMBRs and objects inside of each PMBR;
Output: NN;
Procedure:
1: read PMBR
2: divide RMBR according to the value of $PC_s$ (STEP 1)
3: find C$_q$
4: check N$^c$ // verify if any object is inside of C$_q$
5:   if C$_q$ contains at least one object then
6:       draw safety_circle centered at $q$ which contains C$_q$ (STEP 2)
7:   else find C$_{near}$
8:       draw safety_circle centered at $q$ which contains C$near$ (STEP 3)
9: find C$_{safe}$ and check S$_{safe}$
10: while(obtain NN)
11:  for each object O$_i$
12:      if($x$-coordinate of O$_i$) $\geq$ ($x$-coordinate of C$_{safe}$) AND O $\neq$ O$_{first}$
13:          then
14:          switch to doze mode until S$_{safe}{}^{th+1}$ object arrives // next broadcast cycle (STEP 5)
15:      else
16:          switch to doze mode until S$_{safe}{}^{th+1}$ object arrives // current broadcast cycle (STEP 6)
17:      while (dist (($x$-coordinate of O$_i$), ($x$-coordinate of $q$))$\leq$ dist (NN$_p$, $q$))
18:          if O$_i$ is currently nearest distance from $q$ then O$_i$ = NN$_p$ (STEP 7)
19:      end while
20:      NN$_p$=NN (STEP 8)
21:  end while
22:  return NN
```

Fig. 11. NN Query with Selective Tuning Algorithm

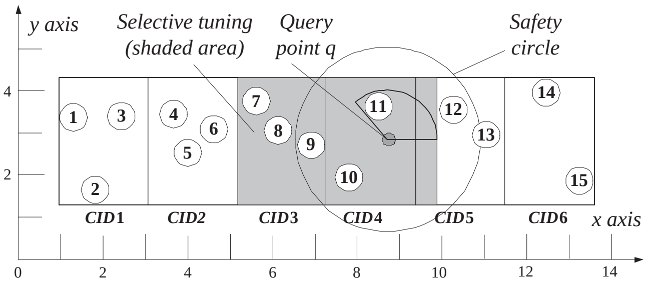


Fig. 12. Selective NN Search

The following lemma ensures the earliest decision time that the client stops tuning and returns the query result.

Lemma 1: Given a query point q, while the data objects are broadcast in horizontal order, if O$_i$ $\neq$ O$_{first}$ and dist ((x-coordinate of O$_i$),(x-coordinate of q)) > dist (NN$_p$, q), then O$_i$ and the objects located in right-side of O$_i$ cannot be NN, where (x-coordinate of O$_i$) > (x-coordinate of NN$_p$) and (x-coordinate of O$_i$) > (x-coordinate of q).

Proof: Since *dist* (NN$_p$, q) $<$ *dist* (O$_i$, q) and *dist* (NN$_p$, q) $<$ *dist* (|(x-coordinate of O$_i$)-(x-coordinate of q)|), if O$_i$ is located in right-side of NN$_p$ and q, then O$_i$ and the objects that are located in right-side of O$_i$, i.e. O$_{i+n}$, are farther from q than NN$_p$, where (x-coordinate of O$_{i+n}$) > (x-coordinate of O$_i$) and (n > 1). □

If the client satisfies the Lemma 1, it returns O$_i$ as the final result of the NN query while it selectively tunes a broadcast channel from the ARD client. This process is illustrated in Figure 12, where objects located in the shaded area are the objects that the client tunes during NN query processing.

Definition 4 defines the cells which do not need to tune during NN query processing.

Definition 4: (selective tuning for NN query). Let C$_i$ denotes a cell which contains object O$_i$, where (x-coordinate of O$_i$) > (x-coordinate of NN$_p$) and (x-coordinate of O$_i$) > (x-coordinate of q) and dist ((x-coordinate of O$_i$),(x-coordinate of q)) > dist (NN$_p$, q). Given a query point q, although the client stays in sleep mode and misses the following cell C$_j$ from PMBR$_i$: (x-coordinate of C$_j$) < (x-coordinate of C$_{safe}$) or (x-coordinate of C$_j$) > (x-coordinate of C$_i$), it can be guaranteed that it does not miss the NN$_p$ from PMBR$_i$.
Thus, even though the client does not tune the objects inside the cells, it can still be guaranteed that it does not miss the object that is the closest to a query point.

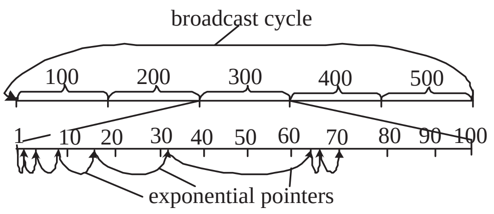


Fig. 13. Selective Tuning with DSS

6.3 Selective Tuning with DSS

As described in Section 2.3, the ESS algorithm provides clients with the ability to perform selective tuning to assist in reducing the client's tuning time. With the exponential index structure, each data item has its location and NP as mentioned in Section 2. In this section, we propose a DSS (Distributed ESS) algorithm. In DSS, the query processing is the same as ESS. However, the size of NP is smaller then ESS since NP is distributed according to PC_s value. That is, NP only has the pointers of potential object(s) that are located inside of each cell. We now suppose that the ARD client periodically broadcasts 500 data items in HB order. Let the PC_s value be 5 and data objects be uniformly distributed in 1D space for simplicity. PMBR has five cells and each cell contains 100 objects. Assume that the client processes the NN query and the NN object is located at the 370th in cell 3 among 500 objects. Then, the client: (i) stays in doze mode until the first data object of cell 3 (the 301th) arrives, (ii) wakes up and reads NP from the 301th object, (iii) switches to doze mode until the 64th object arrives, (iv) wakes up and reads NP from the 64th object, (v) switches to doze mode until the 70th objects arrives, and (vi) wakes up and obtains the final result (see Figure 13 for an example). Let us introduce Definition 5, used for selective tuning with DSS for NN queries.

Definition 5: (selective tuning with DSS). Let O$_{left}$ denotes the currently nearest object on the left-hand side of the q according to NP. While the client selectively tunes the broadcast channel, in order to avoid missing any object(s) which can be the NN (or one of members of the result set for range query), the selective object O$_i$ must satisfy the following condition: dist (x-coordinate of object O$_i$, x-coordinate of q) ≥ dist (O$_{left}$, q), where (x-coordinate of O$_i$) ≤ (x-coordinate of q).

While the client selectively tunes the ARD client's broadcast channel with the DSS, in order to guarantee that it does not miss the target object(s), the selective object O$_i$ must satisfy Definition 5.

Figure 14 shows the pseudo-code for the selective tuning algorithm with DSS.

```
Parameters:
NNQ: NN query processing
RangeQ: range query processing
O_left: currently nearest object on the left-hand side of the q according to NP
O_left': safety nearest object on the left-hand side of the q which satisfies
Definition 5
Input: data objects O_i;
Output: final_result;
Procedure:
1: while (identify O_left' object)
2:    for each object O_i
3:       read NP
4:          find O_left
5:             switch to doze mode until O_left arrives
6:             switch to wake up mode when O_left arrives
7: end while
8: switch to doze mode until O_left' arrives
9: start to read (tune) objects from O_left'
10:  while (obtain the final_result)
11:     for each object O_i
12:        if O_i is inside of q // for RangeQ
13:           then result_set=result_set ∪ O_i
14:        else if O_i=NN_p AND satisfy the Lemma 1 (in order to identify NN)
// for NNQ
15:           then O_i=NN
16:  end while
17: return final_result
```

Fig. 14. Selective Tuning with DSS Algorithm

The DSS algorithm provides a fully distributed structure that allows query processing to start quickly. However, performance may decrease because of the additional information necessary to assign pointers. Let us compare the DSS algorithm with the (1, m) index scheme. Let us assign 16 bytes for each pointer, which also includes 2D coordinates. Let us assign 32 bytes for the (1, m) index since it needs more space to maintain spatial information (e.g. MBR (Minimal Bounding Rectangles)) and pointers for index nodes.

Indexes are interleaved m times among the broadcast data items. Figure 15 shows the access times of DSS and (1, m) index as the size of data items increases. The (1, m) index outperforms DSS when the size of the data items is smaller than 64 bytes. However, DSS outperforms the (1, m) index as the

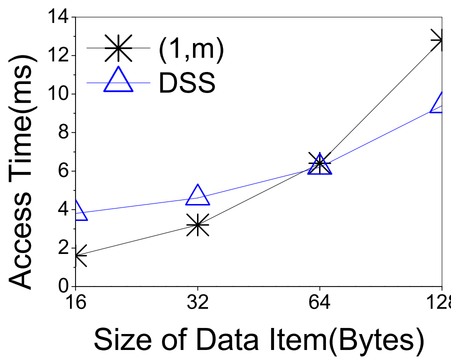


Fig. 15. Access Time as the Size of Data Item Increases

size of data items increases since the increasing data size significantly affects the probe wait time. We conclude that the index overhead caused by the pointer is acceptable when the size of the data items is not quite small. Please refer to [24], [30] for the detailed selective tuning algorithm.

7 PERFORMANCE EVALUATION

In this section, we verify the effectiveness and the efficiency of the proposed PMBR index with the selective tuning algorithm by comparing it with the on-demand approach and with spatial index algorithms, such as R-tree and the Hilbert curve-based air index. We assume that the client's mobility pattern follows the Random Waypoint Mobility Model [31] which is widely used. A mobile client begins by staying in one location for a certain period of time t. We now describe the ratio of energy consumption for these states. The average energy consumption can be measured by the amount of unit energy in a given time. We assume two energy states, *doze mode* and *active mode*. In processors, the doze mode has extremely low power consumption. In the Hobbit chip from AT&T, for example, the ratio of power consumption in the active mode to the doze mode is 5,000 [12]. Let ec be the energy coefficient which means the active-to-doze ratio, $\frac{\mathcal{E}_{ACTIVE}}{\mathcal{E}_{DOZE}}$.

The average energy consumption can be measured by the amount of unit energy in a given time. In order to choose reasonable coefficients, we use various reference values [12], [32]. The energy coefficients ($\widehat{ec}$) can be estimated. In our experiment, parameter ec is fixed with 48.61 as follows.

$$\widehat{ec} \;=\; \frac{\mathcal{E}_{ACTIVE}}{\mathcal{E}_{DOZE}} = \frac{(400+750+462)\ mW}{(0.16+0+33)\ mW} = 48.61 \qquad (5)$$

Table 1 shows the energy consumption measurement model of different states for our experiment [33].

7.1 Performance Analysis

We consider three different data access approaches. On-demand Approach (OA, for short) uses a point-to-point channel for the final result without selective contacting or tuning of nodes. PMBR with hybrid (PMBR_HYD, for short) uses both a broadcast channel for the selective contacting of nodes (using PMBR) and a point-to-point channel for the final result. PMBR with DSS (PMBR_DSS, for short) uses a broadcast channel for both the selective contacting of nodes (using PMBR) and the selective tuning of data items (using DSS). We assume that the channel bandwidth of both the download and the upload (or request) is the same.

TABLE 1
Energy Consumption Model

Technique	$\mu W.sec/byte^7$ $\mu W.sec$
point-to-point send	1.9 $\times$*size* + 454
broadcast send	1.9 $\times$*size* + 266
point-point recv	0.50 $\times$*size* + 356
broadcast recv	0.50 $\times$*size* + 56

Let us define the following notations: (i) Let ε_{recv} and ε_{send} be the energy consumption for P2P receive and P2P send, respectively, (ii) Let d, Req, Ack and s be the packet size for: the sum of downloading all required data items; request message; acknowledge message; and selective tuning of NP pointers, respectively, (iii) Let n be the number of contact nodes until the client identifies the final result, and (iv) Let r be the number of request nodes since the client received n PMBR indexes, where $r \leq n$ (applicable to PMBR_HYD algorithm only).

Let ε_{OA} be the amount of time for energy consumption while the client sends n request messages and receives n acknowledge messages and d data items, Then ε_{OA} is:

$$\frac{Req \times n}{bandwidth} \times \varepsilon_{send} + \frac{Ack \times n + d}{bandwidth} \times \varepsilon_{recv} \qquad (6)$$

Let ε_{PMBR_HYD} be the amount of time for energy consumption while the client sends r request messages and receives n PMBR indexes and d data items, then ε_{PMBR_HYD} is:

$$\frac{Req \times r}{bandwidth} \times \varepsilon_{send} + \frac{PMBR \times n + d}{bandwidth} \times \varepsilon_{recv} \qquad (7)$$

Let ε_{PMBR_DSS} be the amount of time for energy consumption while the client receives n PMBR indexes, s pointers and d data items, then ε_{PMBR_DSS} is:

$$\frac{PMBR \times n + s + d}{bandwidth} \times \varepsilon_{recv} \qquad (8)$$

Now we compare the average access time of DSS with (1, m) index which interleaves m spatial indexes (e.g. R-tree or Hilbert Curve-based). As we mentioned in Section 2, the access time for (1, m) is $\frac{1}{2}\times((m+1)\times i + \frac{1}{m} + 1 \times N) + d$. Since DSS eliminates the probe wait time by sequential broadcasting of data items, the access time for DSS is $\frac{N-d}{2} + d$. Let us compare the tuning time for the DSS algorithm with the (1, m) index. Let k' be the index search cost for (1, m). Then, the tuning time for (1, m) is $1+(k' \times d)+d$. Let e be the exponent value, i' be the size of NP and N^c be the number of objects in the cell. In DSS, the minimum number of steps for querying is 1 and the maximum number of steps is $k-1$, where k=$|log_e N^c|$. For example, if N^c=1024 and e=2, then, in the best case, the client obtains the desired data item within a single step while, in the worst case, the client obtains the desired data item within 9 steps. The frequency of the worst case for N^c is 1, while the frequency of the best case for N^c is k. Thus, the tuning time for DSS is

$$\frac{k \times 2^{k-1} \times \sum_{i=0}^{e-1} i}{N^c} \times i' = \frac{k \times e(e-1)}{4} \times i' \qquad (9)$$

Let AEC be the average energy consumption. Since the average energy consumption is the amount time of active and doze mode, AEC_(1, m) be:

$$\{1+(k' \times d)+d\} \times \varepsilon_{Active} + \{\frac{1}{2} \times ((m+1) \times i + \frac{1}{m} + 1 \times N) + d - 1 + (k' \times d) + d\} \times \varepsilon_{Doze} \qquad (10)$$

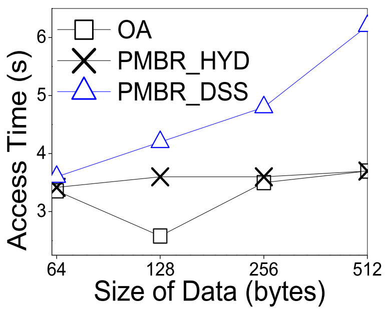
and AEC_DSS be:

$$\{\frac{k \times e(e-1)}{4} \times i'\} \times \varepsilon_{Active} + \{\frac{N-d}{2} + d - \frac{k \times e(e-1)}{4} \times i'\} \times \varepsilon_{Doze} \qquad (11)$$

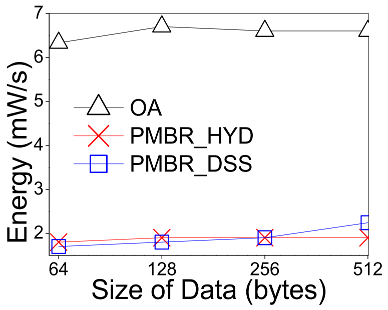
In summary, OA significantly increases the communication overhead among the nodes and causes extremely high energy consumption, especially when the nodes are crowded in specific areas, such as a university or train station. Moreover, the delay time of a query increases as the number of query request messages from neighbor nodes increases. The proposed DSS algorithm significantly reduces energy consumption compared with the (1,m) index scheme, since it minimizes both the access time and the tuning time. In other words, when we estimate the energy consumption, we must consider not only the active time, but also the idle time, even though the energy consumption in the idle mode is lower than that in the active mode.

7.2 Experiments

In this section, we evaluate the efficiency of our algorithms with a Pentium 3.0 GHz CPU. We assume that the broadcast channel has a bandwidth of 2 Mbps. We use a UNIFORM dataset (hereafter called $\mathcal{D}1$) with 120,000 points, and a REAL dataset (hereafter called $\mathcal{D}2$) containing the 5922 data objects of cities and villages of Greece, extracted from the data set available at www.rtreeportal.org [34] (see Figures 16(a) and 16(b)). We assign 16 bytes for each pointer which also includes 2D coordinates. We assign 32 bytes for the spatial index since it needs more space to maintain spatial data and pointers in the index. Indexes are interleaved m times among the broadcast data items. The client broadcasts data items in horizontal order based on their locations. The default parameter setting in our synthetic data set test is: number of objects = 10,000, Size of data = 128 bytes, e = 2, moving distance = 100 (m). The first set of experiments examines $\mathcal{AT}$ (access



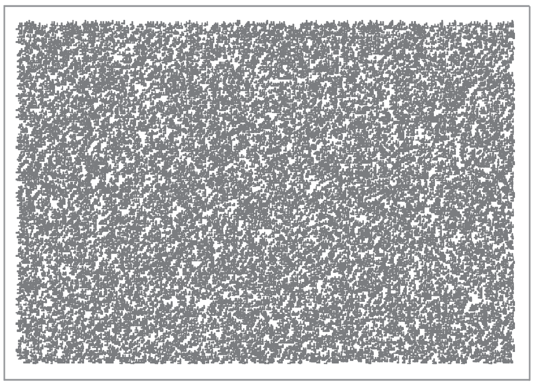
(a)



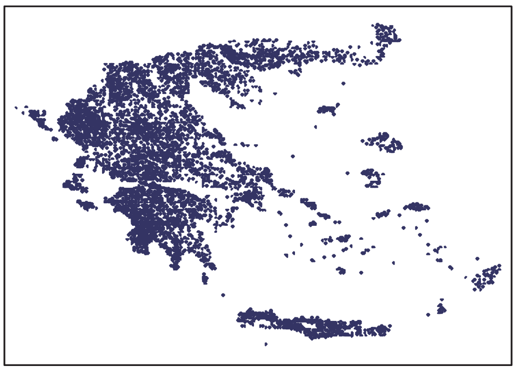
(b)

Fig. 17. Effect of Size of Data on NN Query in $\mathcal{D}2$. (a) Access time. (b) Energy consumption.

time) for NN queries as the size of data items increases in $\mathcal{D}2$. As shown in Figure 17(a), OA and PMBR_HYD outperform PMBR_DSS since the client of PMBR_DSS must wait until the desired objects arrive in the broadcast channel, while the client of OA and PMBR_HYD directly requests and obtains the desired objects from the nodes. However, for $\mathcal{EC}$ (energy consumption), as shown in Figure 17(b), PMBR_HYD and PMBR_DSS clearly outperform OA since the client of OA must repeatedly send the request messages to each neighbor node whether that node has the desired information or not. The clients of PMBR_HYD and PMBR_DSS send a request message (or selectively tune) only to the node that has the desired objects according to the PMBR index. Figure 18(a) and 18(b) illustrate the $\mathcal{AT}$ and the $\mathcal{EC}$ for range queries as the size of data items increases in $\mathcal{D}2$. Similarly, for $\mathcal{AT}$, OA and PMBR_HYD outperform PMBR_DSS, but for $\mathcal{EC}$, PMBR_HYD and PMBR_DSS outperform OA. Figure 18(b) shows that PMBR_DSS outperforms PMBR_HYD when the size of the data items is smaller than 128 bytes. Then, PMBR_HYD

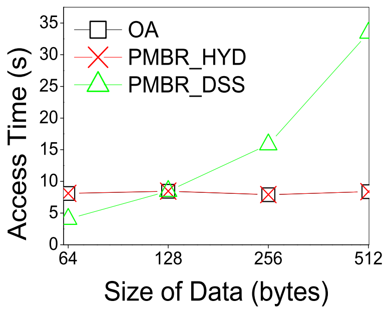


(a)

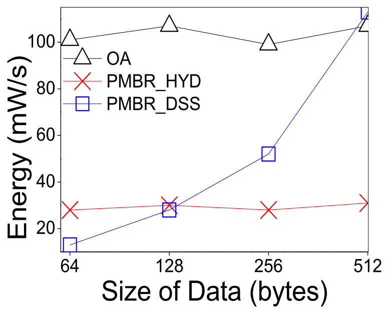


(b)

Fig. 16. Datasets for Performance Evaluation. (a) UNIFORM dataset. (b) REAL dataset.

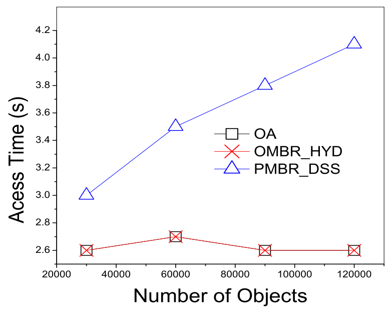


(a)

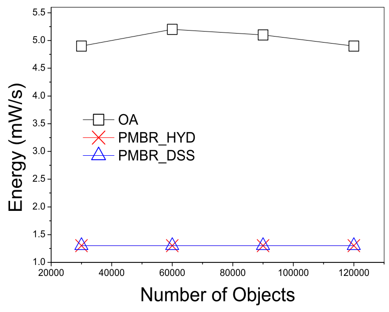


(b)

Fig. 18. Effect of Size of Data on Range Query in $\mathcal{D}2$. (a) Access time. (b) Energy consumption.



(a)



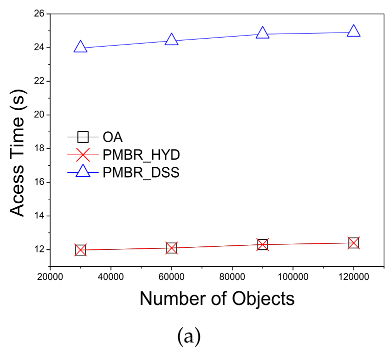
(b)

Fig. 19. Effect of Number of Objects on NN Query in $\mathcal{D}1$. (a) Access time. (b) Energy consumption.

starts to gradually outperform PMBR_DSS since the average time for the selective tuning of PMBR_DSS is increased as the size of data items increases. Figure 19(a), 19(b), 20(a), and 20(b) illustrate the $\mathcal{AT}$ and the $\mathcal{EC}$ for NN and range queries as the number of data objects increases in $\mathcal{D}1$, respectively. Similar results are obtained for NN and range queries. From the results, we can observe that the number of objects almost does not affect the $\mathcal{AT}$ and the $\mathcal{EC}$ since the clients contact (or tune) desired data items from the near nodes based on their current locations.

Figure 21(a), 21(b), 22(a), and 22(b) illustrate the $\mathcal{AT}$ and the $\mathcal{EC}$ for NN and range queries as the client's moving distance increase in $\mathcal{D}2$, respectively. As shown in the figures, the moving distance of the client affects both the $\mathcal{AT}$ and the $\mathcal{EC}$. The nearest object (or objects in the range) may change while the client continuously moves. The client continuously contacts (or tunes) the near nodes in order to obtain the correct answer. In case of the $\mathcal{AT}$, OA and PMBR_HYD outperform PMBR_DSS, whereas in case of the $\mathcal{EC}$, PMBR_DSS and PMBR_HYD outperform OA for the same reason as above.

Figure 23(a) and 23(b) illustrate the $\mathcal{AT}$ and the $\mathcal{EC}$ for NN queries as the cache size of the nodes increases in $\mathcal{D}2$, respectively. As shown in the figures, as expected, the $\mathcal{AT}$ and the $\mathcal{EC}$ decrease drastically as the cache size of the node increase. The reason



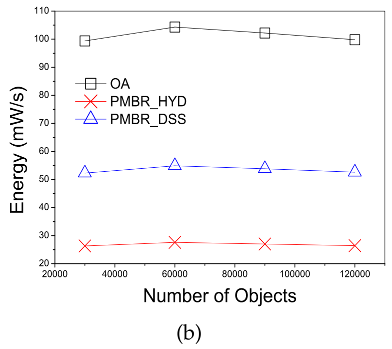
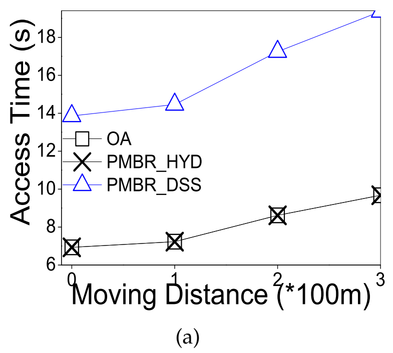


Fig. 20. Effect of Number of Objects on Range Query in $\mathcal{D}1$. (a) Access time. (b) Energy consumption.



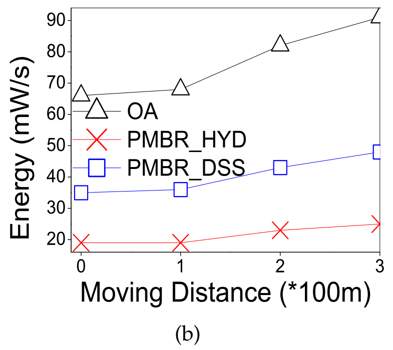
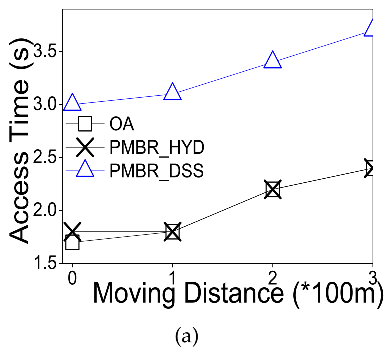


Fig. 22. Effect of Moving Distance on Range Query in $\mathcal{D}2$. (a) Access time. (b) Energy consumption.



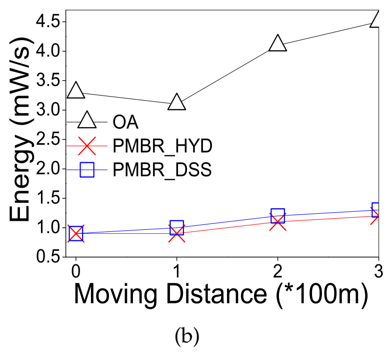
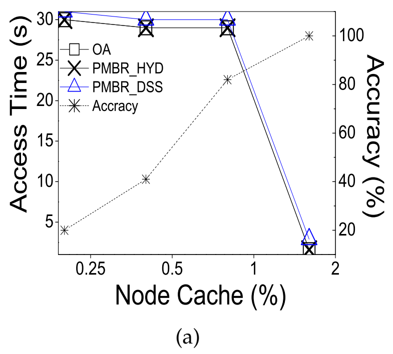


Fig. 21. Effect of Moving Distance on NN Query in $\mathcal{D}2$. (a) Access time. (b) Energy consumption.



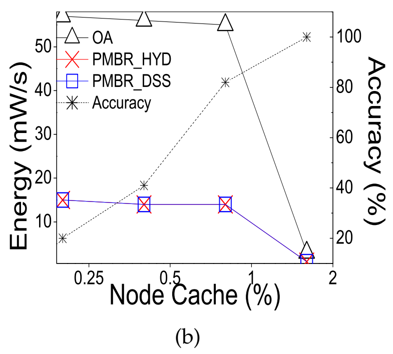
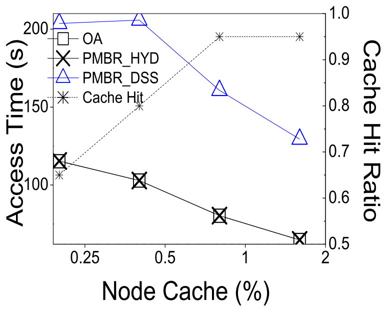
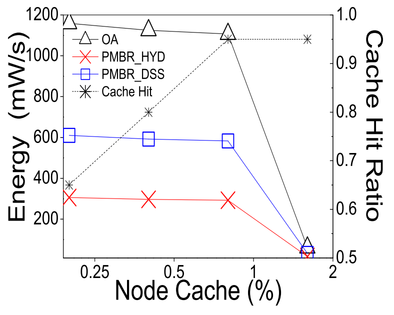


Fig. 23. Effect of Cache Size on NN Query in $\mathcal{D}2$. (a) Access time and accuracy for the result. (b) Energy consumption and accuracy for the result.

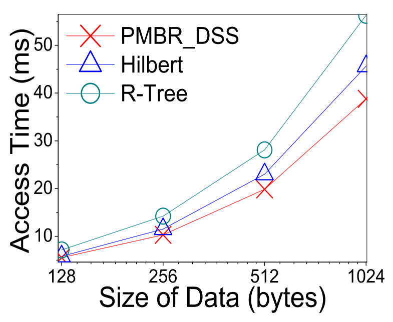


(a)

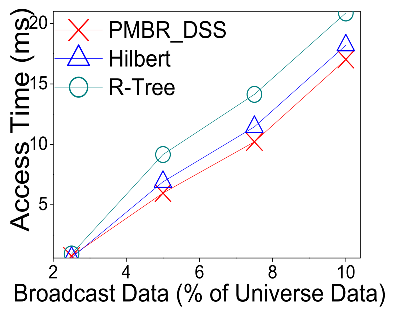


(b)

Fig. 24. Effect of Cache Size on Range Query in $\mathcal{D}2$. (a) Access time and cache hit ratio. (b) Energy consumption and cache hit ratio.



(a)

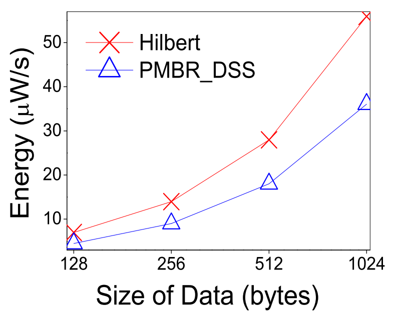


(b)

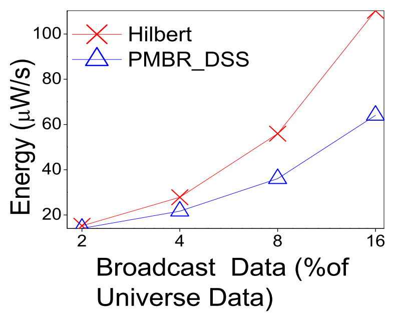
Fig. 25. Effect of size and the number of data on NN Query. (a) Access time as the size of data increases. (b) Access time as the % of universe data increases.

for this is that the nodes are able to answer most of the client's NN queries if they have adequate cache capacity. The accuracy for the result of NN queries also increases as the cache size of nodes increases. Notice that the client is unable to find the nearest object if the nodes do not have the desired object in the cache. In this case, the client returns the near object instead of the nearest object from its location. Figures 24(a) and 24(b) show the $\mathcal{AT}$ and the $\mathcal{EC}$ for range queries as the cache size of nodes increases in $\mathcal{D}2$. Clearly, the $\mathcal{AT}$ and the $\mathcal{EC}$ decrease as the cache size of nodes increases while the cache hit ratio increases.

Next, we evaluate the $\mathcal{AT}$ and the $\mathcal{EC}$ of the PMBR_DSS with R-tree and Hilbert-curve-based air index. We present NN queries as representative of spatial indexes (the result of a range query is almost the same as the NN queries). The client which processes NN queries tunes the ARD client's broadcast channel and predicts the approximate arrival time of the desired data items using the index, i.e. PMBR, Hilbert or R-tree. Figures 25(a) and 25(b) show the $\mathcal{AT}$ as the size and the number of data items (% of the universe) increase. As shown in the figures, PMBR_DSS outperforms R-tree and Hilbert index and provides the optimal access time. The main reason for this is that the proposed algorithm eliminates the probe wait time. The DSS algorithm is a fully distributed structure that allows query processing to start quickly. Figures 26(a) and 26(b) show the $\mathcal{EC}$



(a)



(b)

Fig. 26. Effect of size and the number of data on NN Query. (a) Energy consumption as the size of data increases. (b) Energy consumption as the % of universe data increases.

as the size and the number of data items (% of the universe) increase. In this experiment, we compare PMBR_DSS with a Hilbert-curve-based index only since the R-tree index performs much worse than the Hilbert-curve-based index in wireless data broadcast environments. As shown in the figures, PMBR_DSS outperforms the Hilbert-curve index. This happens because PMBR_DSS simultaneously reduces both the "tuning time" by selective tuning and the "access time" by eliminating probe wait time. In other words, when the energy consumption is estimated, it is necessary to consider the active time as well as the idle time. In summary, the experiments confirm that our approach helps reduce the average access time and the energy consumption since the data objects broadcast by the ARD client are sequentially ordered based on their locations. It is not necessary for the client to wait for an index segment, thus eliminating probe wait time.

8 Conclusion

In this paper, we proposed energy efficient data access algorithms for range and NN queries, specifically designed for mobile P2P environments. First, we proposed a PMBR index that provides the client with the ability to selectively contact and tune from other nodes. The mobile client is able to identify whether or not the neighbor nodes have the desired information by accessing the index. Thus, the client immediately switches to change to another node's broadcast channel if the index does not contain the desired information. Furthermore, the client is able to predict the arrival time of the desired data items and only needs to tune into the broadcast channel when the requested data items arrives, by first accessing the broadcast index. Thus, the mobile client can stay in power save mode most of the time, and tune into the broadcast channel only when the requested data items arrive. PMBR is partitioned according to the value of PC_s in order to support efficient selective tuning and obtain more accurate results. Second, we proposed a new selective tuning algorithm, called DSS, for the selective tuning of spatial queries. With DSS, the client can process spatial queries without needing index information for selective tuning.

We proved, both theoretically and empirically, that PMBR with our selective tuning algorithm provides accurate answers with minimal access latency and energy consumption. The result of performance evaluation showed that PMBR_DSS gives the best performance when the number and the size of the data items are small and is suitable when many users share the same interest. PMBR_HYD gives the best performance when the number and the size of the data items are large but few users share the same interest.

Our solution (i.e. PMBR and DSS) yields the following advantages: (i) It incurs small traffic overhead among the nodes. (ii) It facilitates energy conservation by reducing request messages and providing the ability to perform selective tuning. (iii) It provides fast response time since the client only sends requesting messages to the node with the desired data items. (iv) It can be easily extended to other types of queries, such as k-nearest neighbor queries and continuous spatial queries.

In the future, we plan to study the influence of user's movement on location-dependent caching strategies based on the observation that the result of queries may change as the user or target object changes its location. Also, we intend to extend the proposed algorithms to other query types, such as *dynamic range query* and *continuous k-nearest neighbor query* in a wireless data broadcast environment.

Acknowledgment

This paper was supported by Wonkwang university in 2010.

References

[1] H. V. Jagadish, B. C. Ooi, Q. H. Vu, R. Zhang and A. Zhou, "VBI-Tree: A Peer-to-Peer Framework for Supporting Multi-Dimensional Indexing Schemes, " In *Proc. of International Conference on Data Engineering (ICDE)*, 2006, pp 34.

[2] B. Xu and O. Wolfson, "Data Management in Mobile Peer-to-Peer Networks," In *Proc. of Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)* 2004, pp. 1-15.

[3] Z. Huang, C. S. Jensen, H. Lu and B. C. Ooi, "Skyline Queries Against Mobile Lightweight Devices in MANETs," In *Proc. of International Conference on Data Engineering (ICDE)* 2006, pp. 66.

[4] H. V. Jagadish, B. C. Ooi and Q. H. Vu, "BATON: A Balanced Tree Structure for Peer-to-Peer Networks," In *Proc. of Very Large Data Base (VLDB)*, 2005, pp. 661-672.

[5] K. Aberer, A. Datta, M. Hauswirth and R. Schmidt, "Indexing Data-oriented Overlay Networks," In *Proc. of Very Large Data Base (VLDB)*, 2005:, pp. 685-696.

[6] J. X. Parreira, D. Donato, S. Michel and G. Weikum, "Efficient and Decentralized PageRank Approximation in a Peer-to-Peer Web Search Network," In *Proc. of Very Large Data Base (VLDB)*, 2006, pp. 415-426.

[7] K. C. K. Lee, W.-C. Lee, J. Winter, B. Zheng and J. Xu, "CS cache engine: data access accelerator for location-based service in mobile environments," In *Proc. of Special Interest Group on Management of Data (SIGMOD)*, 2006, pp. 787-789.

[8] D. L. Lee, W.-C. Lee, J. Xu, and B. Zheng, "Data Management in Location-dependent Information Services: Challenges and Issues," *IEEE Pervasive Computing*, 2002, 1(3), 65-72.

[9] J. Zhang, M. Zhu, D. Papadias, Y. Tao and Dik Lun Lee, "Location-based Spatial Queries,", In *Proc. of Special Interest Group on Management of Data (SIGMOD)*, 2003, pp. 443-454.

[10] B. Zheng, J. Xu, W.-C. Lee and D. L. Lee, "Grid-partition index: a hybrid method for nearest-neighbor queries in wireless location-based services," In *Very Large Data Base Journal (VLDB J)*, 2006, 15(1), 21-39.

[11] B. Zheng, W.-chien Lee and D. L. Lee, "Spatial Queries in Wireless Broadcast Systems,"*Wireless Network (WINET)*, 2004, 10(6), 723-736.

[12] T. Imielinski, S. Viswanathan, and B.R.Badrinath, "Data on Air: Organization and Access," *IEEE Trans. Knowledge and Data Eng (TKDE)*, 1997, 9(3), 353-372.

[13] T. Imielinski, S. Viswanathan, and B.R.Badrinath, "Energy efficient indexing on air," In *Proc. of Special Interest Group on Management of Data (SIGMOD)*, 1994, pp. 25-36.
[14] R. Zimmermann, W.S. Ku and H. Wang, "Spatial Data Query Support in Peer-to-Peer Systems," In *Proc. of International Computer Software and Applications Conference (COMPSAC) Workshops*, 2004, pp. 82-85.
[15] H. Wang, R. Zimmermann and W.S. Ku, "ASPEN, an adaptive spatial peer-to-peer network," In *Proc. of Geographic Information Systems (GIS)*, 2005, pp. 230-239.
[16] O. Wolfson, B. Xu and H. Yin, "Dissemination of Spatial-Temporal Information in Mobile Networks with Hotspots," In *Proc. of Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, 2004, pp. 185-199.
[17] A. Mondal, Y. Lifu and M. Kitsuregawa, "P2PR-Tree: An R-Tree-Based Spatial Index for Peer-to-Peer Environments," In *Proc. of Extending Database Technology (EDBT) Workshops*, 2004, pp. 516-525.
[18] W.-S. Ku, R. Zimmermann, C.-N. Wan, and H. Wang, "MAPLE: A Mobile Scalable P2P Nearest Neighbor Query System for Location-based Services," In *Proc. of International Conference on Data Engineering (ICDE)*, 2006, pp. 160.
[19] M. Demirbas and H. Ferhatosmanoglu, "Peer-to-Peer Spatial Queries in Sensor Networks," In *Proc. of International Conference on Peer-to-Peer Computing*, 2003, pp. 32-39.
[20] G. Monti and G. Moro, "Multidimensional Range Query and Load Balancing in Wireless Ad Hoc and Sensor Networks," In *Proc. of International Conference on Peer-to-Peer Computing*, 2008, pp. 205-214.
[21] P. Zhou, T. Nadeem, P. Kang and C. Borcea, "Liviu Iftode: EZCab: A Cab Booking Application Using Short-Range Wireless Communication," In *Proc. of Pervasive Computing and Communications (PerCom)*, 2005, pp. 27-38.
[22] A. Guttman, "R-trees: A dynamic index structure for spatial searching," In *Proc. of Special Interest Group on Management of Data (SIGMOD)*, 1984, pp. 47-57.
[23] K. Park and C.-S. Hwang, "Client-Side Caching for Nearest Neighbor Queries," *Journal of Communication and Networks (JCN)*, 2005, 7(4), 417-428.
[24] K. Park, M. Song and C.-S. Hwang, "Continuous spatial queries via wireless data broadcast," In *Symposium on Applied Computing (SAC)*, 2006, pp. 78-82.
[25] K. Park and H. Choo, "Energy-Efficient Data Dissemination Schemes for Nearest Neighbor Query Processing," IEEE Trans. Computers (TC), 2007, 56(6), 754-768.
[26] K. Park, P. Valduriez, and H. Choo, "Mobile continuous nearest neighbor queries on air," In *ACM SIGSPATIAL international conference on Advances in geographic information systems (ACM-GIS)*, 2008, pp. 65.
[27] S.E. Hambrusch, C.-M. Liu,W. Aref, and S. Prabhakar, "Query processing in broadcasted spatial index trees," In *Proc. of Symposium on Spatial and Temporal Databases (SSTD)*, 2001, pp. 502-521.
[28] A. Prasad Sistla, O. Wolfson and B. Xu, "Opportunistic Data Dissemination in Mobile Peer-to-Peer Networks," In *Proc. of Symposium on Spatial and Temporal Databases (SSTD)*, 2005, pp. 346-363.
[29] N. Roussopoulos, S. Kelley and F. Vincent, "Nearest Neighbor Queries," In *Proc. of Special Interest Group on Management of Data (SIGMOD)*, 1995, pp. 71-79.
[30] K. Park, M. Song, K.-S. Kong, S.-W. Kang, C.-S. Hwang, K.-S. Chung and S.Y. Jung, "Effective Low-Latency K-Nearest Neighbor Search Via Wireless Data Broadcast," In *Proc. of Database Systems for Advanced Applications (DASFAA)*, 2006, pp. 900-909.
[31] T. Camp, J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication and Mobile Computing (WCMC)*, 2002, 2(5), 483-502.
[32] O. Kasten, "Energy consumption," ETH-Zurich, Swiss Federal Institute of Technology. Available at http://www.inf.ethz.ch/~kasten/research/bathtub/energy_consumption.html.
[33] L. M. Feeney and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," *IEEE Conference on. Computer Communications (INFOCOM)*, 2001, pp. 1548-1557.
[34] http://www.rtreeportal.org

PLACE PHOTO HERE

Kwangjin Park Kwangjin Park received the Ph.D. degree in Computer Science from Korea University in 2006. He was a post-doctoral fellow in the Atlantic Data Systems (Atlas) Research Group at the Institut National de Recherche en Informatique et en Automatique (INRIA)-Rennes and located at the Laboratoire dinformatique de Nantes-Atlantique (LINA), Nantes. In 2008, he joined the Schools of Electrical Electronics and Information Engineering, Wonkwang University, where he is an assistant professor. His research interests include spatiotemporal databases, mobile databases, and data dissemination.

PLACE PHOTO HERE

Patrick Valduriez Patrick Valduriez is a Director of Research at INRIA, the national research center for computer science in France and scientific director of the Atlas research group pursuing research in data management in distributed systems. He received his Ph. D. degree and Doctorat d'Etat in Computer Science from the University Pierre et Marie Curie (Paris 6) in 1981 and 1985, respectively.

In 1999-2002, he was a professor of computer science at University Paris 6, on leave from INRIA. From 1989 to 1999, he led the Rodin group (20 people) working on distributed database and multimedia technology. From 1995 to 2000, he also led Dyade, the R&D joint venture between Bull and INRIA to foster technology transfer in the areas of Internet (50 people). In Dyade, he contributed to the design and the transfer of the Disco prototype to the KelKoo startup. From 1985 to 1989, he was a senior scientist at Microelectronics and Computer Technology Corp. (MCC) in Austin, Texas. There he participated in the design and implementation of the Bubba parallel database system.

He has authored and co-authored over 200 technical papers and several books; among which, "Principles of Distributed Database Systems" published by Prentice Hall in 1991 and 1999 (2nd edition), "Object Technology" published by Thomson Computer Press in 1997, and Relational Databases and Knowledge Bases published by Addison-Wesley in 1990.

He has been a member of the SIGMOD board, a trustee of the VLDB endowment and an associate editor of several journals; including ACM Transactions on Database Systems, the VLDB Journal, Distributed and Parallel Databases, Internet and Databases: Web Information Systems, etc. He has also served as program chair of major conferences such as PDIS'93, SIGMOD'97, VLDB'98 (industrial program), VLDB'99 (European program). He was the general chair of the ACM SIGMOD/PODS04 conference in Paris (first time outside North America) and of EBBT08 in Nantes and VLDB09 in Lyon.

He was the recipient of the 1993 IBM scientific prize in computer science in France. He obtained the best paper award at the VLDB2000 conference