Location-Based Spatial Query Processing in Wireless Broadcast Environments

Wei-Shinn Ku, *Member*, *IEEE*, Roger Zimmermann, *Senior Member*, *IEEE*, and Haixun Wang, *Member*, *IEEE*

Abstract—Location-based spatial queries (LBSQs) refer to spatial queries whose answers rely on the location of the inquirer. Efficient processing of LBSQs is of critical importance with the ever-increasing deployment and use of mobile technologies. We show that LBSQs have certain unique characteristics that the traditional spatial query processing in centralized databases does not address. For example, a significant challenge is presented by wireless broadcasting environments, which have excellent scalability but often exhibit high-latency database access. In this paper, we present a novel query processing technique that, though maintaining high scalability and accuracy, manages to reduce the latency considerably in answering LBSQs. Our approach is based on peer-to-peer sharing, which enables us to process queries without delay at a mobile host by using query results cached in its neighboring mobile peers. We demonstrate the feasibility of our approach through a probabilistic analysis, and we illustrate the appeal of our technique through extensive simulation results.

Index Terms—Broadcast disks, mobile computing, mobile environments, location dependent and sensitive.

1 Introduction

Spatial query processing is becoming an integral part of many new mobile applications. Recently, there has been a growing interest in the use of location-based spatial queries (LBSQs), which represent a set of spatial queries that retrieve information based on mobile users' current locations [2], [29].

User mobility and data exchange through wireless communication give LBSQs some unique characteristics that the traditional spatial query processing in centralized databases does not address. Novel query processing techniques must be devised to handle the following new challenges:

1. **Mobile query semantics**. In a mobile environment, a typical LBSQ is of the form "*find the top-three nearest hospitals*." The result of the query depends on the location of its requester. Caching and sharing of query results must take into consideration the location of the query issuer.
2. **High workload**. The database resides in a centralized server, which typically serves a large mobile user community through wireless communication. Consequently, bandwidth constraints and scalability become the two most important design concerns of LBSQ algorithms [2].
3. **Query promptness and accuracy**. Due to user mobility, answers to an LBSQ will lose their relevancy if there is a long delay in query processing or in communication. For example, answers to the query "*find the top-three nearest hospitals*" received after 5 minutes of high-speed driving will become meaningless. Instead, a prompt, albeit approximate, answer, telling the user right away the *approximate* top-three nearest hospitals, may serve the user much better. This is an important issue, as a long latency in a high workload wireless environment is not unusual.

The wireless environment and the communication constraints play an important role in determining the strategy for processing LBSQs. In the simplest approach, a user establishes a point-to-point communication with the server so that her queries can be answered on demand. However, this approach suffers from several drawbacks. First, it may not scale to very large user populations. Second, to communicate with the server, a client must most likely use a fee-based cellular-type network to achieve a reasonable operating range. Third, users must reveal their current location and send it to the server, which may be undesirable for privacy reasons [19]. A more advanced solution is the wireless broadcast model [1], [15], [30]. It can support an almost-unlimited number of mobile hosts (MHs) over a large geographical area with a single transmitter. With the broadcast model, MHs do not submit queries. Instead, they tune in to the broadcast channel for information that they desire. Hence, the user's location is not revealed. One of the limitations of the broadcast model is that it restricts data access to be sequential. Queries can only be fulfilled after all the required on-air data arrives. This is why in some cases, a 5-minute delay to the query "*find the top-three nearest hospitals*" would not be unusual.

Alleviating this limitation, we propose a scalable low-latency approach for processing LBSQs in broadcast

- *W.-S. Ku is with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849. E-mail: weishinn@auburn.edu.*
- *R. Zimmermann is with the Department of Computer Science, National University of Singapore, Singapore 117590. E-mail: rogerz@comp.nus.edu.sg.*
- *H. Wang is with the IBM T.J. Watson Research Center, Hawthorne, NY 10532. E-mail: haixun@us.ibm.com.*

Manuscript received 6 Mar. 2007; revised 8 Oct. 2007; accepted 3 Dec. 2007; published online 26 Dec. 2007.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2007-03-0067. Digital Object Identifier no. 10.1109/TMC.2007.70780.

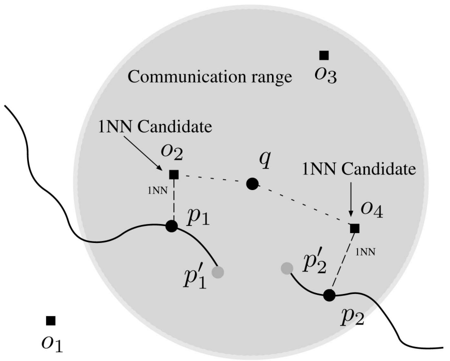


Fig. 1. NN P2P result sharing.

environments. Our approach leverages ad hoc networks to share information among mobile clients in a peer-to-peer (P2P) manner [17], [18]. The rationale behind our approach is based on the following observations:

- As mentioned previously, when a mobile user launches a nearest neighbor (NN) query, in many situations, she would prefer an approximate result that arrives with a short response time rather than an accurate result with a long latency.
- The results of spatial queries often exhibit spatial locality. For example, if two MHs are close to each other, the result sets of their spatial queries may overlap significantly. Query results of a mobile peer are valuable for two reasons: 1) they can be used to answer queries of the current MH directly and 2) they can be used to dramatically reduce the latency for the current MH relative to on-air information.
- P2P approaches can be valuable for applications where the response time is an important concern. Through mobile cooperative caching [7] of the result sets, query results can be efficiently shared among mobile clients.

An example is shown in Fig. 1. At a given time instance, an MH q can establish contact with two other MHs within its communication range: p'_1 and p'_2. In the past, both p'_1 and p'_2 executed NN queries for a certain type of point of interest (POI) when they were located at p_1 and p_2, respectively.[1] The results that they obtained and cached are $< o_2, p_1 >$ and $< o_4, p_2 >$. These two tuples represent candidate solutions for q's own 1NN query. Through a local verification process, q can determine whether one of the solutions obtained from its neighbors is indeed its own nearest POI. Note that the current locations of the neighboring hosts p'_1 and p'_2 have no specific significance, as long as they are within the communication range of q.

In this paper, we concentrate on two common types of spatial searches, namely, kNN queries and window queries (WQs). The contributions of our study are given as follows:

1. We identify certain characteristics of LBSQs that enable the development of effective sharing methods in broadcast environments.
2. We introduce a set of algorithms that verify whether data received from neighboring clients are complete, partial, or irrelevant answers to the posed query.
3. We utilize a P2P-based sharing method to improve the current approaches in answering on-air kNN queries and WQs.
4. We evaluate our approach through a probabilistic analysis of the hit ratio in sharing. In addition, through extensive simulation experiments, we evaluate the benefits of our approach with different parameter sets.

1. In our notation, we use the object identifier to represent its position coordinates.

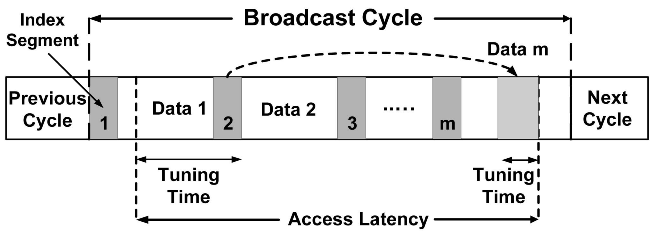


Fig. 2. The data and index organization of the (1, m) indexing scheme with sample tuning time and access latency.

The rest of this paper is structured as follows: Section 2 surveys the related work of the wireless broadcast model, spatial queries, and cooperative caching. Our own approach is detailed in Section 3, and the experimental results are presented in Section 4. Finally, Section 5 concludes this paper and outlines future research directions.

2 Background and Related Work

In this section, we introduce some background information with respect to the support of spatial queries in a wireless broadcast system.

2.1 Wireless Data Broadcast

In general, there are two approaches for mobile data access. One is the *on-demand access model*, and the other is the *wireless broadcast model*. For the on-demand access model, point-to-point connections are established between the server and the mobile clients, and the server processes queries that the clients submit on demand. For the wireless broadcast model, the server repeatedly broadcasts all the information in wireless channels, and the clients are responsible for filtering the information. An example of such a system is the Microsoft DirectBand Network. The advantage of the broadcast model over the on-demand model is that it is a scalable approach. However, the broadcast model has large latency, as clients have to wait for the information that they need in a broadcasting cycle. If a client misses the packets that it needs, it has to wait for the next broadcast cycle.

To facilitate information retrieval on wireless broadcast channels, the server usually transmits an index structure, along with data objects. A well-known broadcast index structure is the (1, m) indexing allocation method [15]. As we can see in Fig. 2, the whole index is broadcast preceding every $1/m$ fraction of the data file. Because the index is available m times in one cycle, it allows a mobile client easy access to the index so that it can predict the arrival time of its desired data in a timely manner, and once it knows the

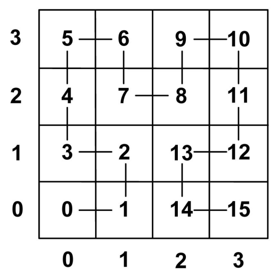


Fig. 3. The Hilbert-curve-based index structure. The numbers represent index values.

arrival time, it only needs to tune into the broadcast channel when the data bucket arrives. This mechanism is important for battery-based devices.

Thus, the general access protocol for retrieving data on a wireless broadcast channel involves three main steps [15]:

- **The initial probe**. A client tunes in to the broadcast channel and determines when the next index segment will be broadcast.
- **Index search**. The client accesses a sequence of pointers in the index segment to figure out when to tune in to the broadcast channel to retrieve the required data.
- **Data retrieval**. The client tunes in to the channel when packets containing the required data arrive and then downloads all the required information.

Two parameters, *access latency* and *tuning time*, characterize the broadcast model. The access latency represents the time duration from the point that a client requests its data to the point that the desired data is received. The tuning time is the amount of time spent by a client listening to the broadcast channel, which proportionally represents the power consumption of the client [15].

However, nearly all the existing spatial access methods are designed for databases with random access disks. These existing techniques cannot be used effectively in a wireless broadcast environment, where only sequential data access is supported. Zheng et al. [31] proposed indexing the spatial data on the server by a space-filling curve. The Hilbert curve [16] was chosen for this purpose because of its superior locality. The index values of the data packets represent the order in which these data packets are broadcast. For example, the Hilbert curve in Fig. 3 tries grouping data of close values so that they can be accessed within a short interval when they are broadcast sequentially. The MHs use on-air search algorithms [31] to answer LBSQs (kNN and WQs) over data that arrives in the order prescribed by the Hilbert curve.

2.2 Spatial Queries

We focus on two common types of spatial queries, namely, kNN queries and WQs. With R-tree-based [10] spatial indices, depth-first search (DFS) [25] and best first search (BFS) [13] have been the prevalent branch-and-bound techniques for processing NN queries. The DFS method recursively expands the index nodes for searching NN candidates. At each newly visited nonleaf node, DFS computes the ordering metrics for all its child nodes and applies pruning strategies to remove unnecessary branches. When a leaf node is reached, the data objects are retrieved, and the NN candidates are updated. Comparatively, the BFS technique utilizes a priority queue to store nodes to be explored through the search process. The nodes in the queue are sorted according to their minimum distance (MINDIST) to the query point. During the search process, BFS repeatedly dequeues the top entry in the queue and enqueues its child nodes with their MINDIST into the queue. When a data entry is dequeued, it is inserted into the result set.

For WQs that find objects within a specified area, the R-tree families [3], [26] provide efficient access to disk-based databases. Basically, an R-tree structure groups objects close to each other into a minimum bounding rectangle (MBR), and a range query only visits MBRs that overlap with the query area.

2.3 Cooperative Caching

Caching is a key technique to improve data retrieval performance in widely distributed environments [14], [21], [22]. Hara and Madria proposed three data replica allocation methods in ad hoc networks by considering the access frequency from MHs to each data item and the status of the network connection [12]. With the increasing deployment of new P2P wireless communication technologies (for example, IEEE 802.11b/g and Bluetooth), *P2P cooperative caching* becomes an effective sharing alternative [6], [11], [28]. With this technique, MHs communicate with neighboring peers in an ad hoc manner for information sharing instead of relying solely on the communication between remote information sources. Yin and Cao [28] proposed three schemes, that is, CachePath, CacheData, and HybridCache, for cooperative caching in ad hoc networks. With CachePath, mobile nodes cache the data path and use it to redirect prospective requests to a neighboring node, which has the data instead of fetching data from the remote data center. With CacheData, intermediate nodes cache the data to serve prospective queries. The HybridCache approach further improves performance by taking advantage of both CacheData and CachePath while avoiding their weaknesses. P2P cooperative caching can bring about several distinctive benefits to a mobile system: improved access latency, reduced server workload, and alleviated point-to- point channel congestion. In this research, we leverage the P2P caching technique to alleviate the inherent access latency limitation in wireless broadcast environments.

3 System Design

In this section, we describe our approach for supporting LBSQs in a wireless broadcast environment. The fundamental idea behind our methodology is to leverage the cached results from prior spatial queries at reachable MHs for answering future queries at the local host.

3.1 Overview

The wireless data broadcast model has good scalability for supporting an almost-unlimited number of clients [15]. Its main limitation lies in its sequential data access: the access latency becomes longer as the number of data items increases. If we can provide (approximate) answers to

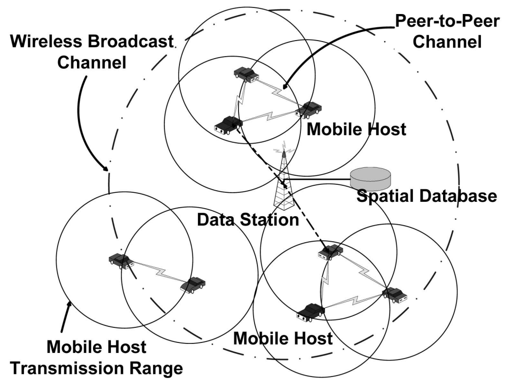


Fig. 4. System environment.

spatial queries before the arrival of related data packets, we will overcome the limitation of the broadcast model.

A novel component in our methodology is a verification algorithm that verifies whether a data item from neighboring peers is part of the solution set to a spatial query. Even if the verified results constitute only part of the solution set, in which case the query client needs to wait for the required data packets to get the remaining answers, the partial answer can be utilized by many applications that do not need exact solutions but require a short response time (for example, the query "What are the top-three nearest hospitals?" issued by a motorist on a highway).

In this study, we detail how kNN queries and WQs can be processed by cooperating MHs to improve the performance of on-air spatial queries (OASQs). We apply the spatial query algorithms proposed in [31] to illustrate our techniques. However, our sharing-based solution can be a common method for any broadcast system.

3.2 Assumed Infrastructure

Fig. 4 depicts our operating environment with two main entities: a remote wireless information server and MHs. We are considering mobile clients such as vehicles, which are instrumented with global positioning systems (GPSs) for continuous position information. Furthermore, we assume that the wireless information server broadcasts information in a wireless channel periodically and the channel is open to the public. In addition, there are short-range networks that allow ad hoc connections with neighboring mobile clients. Technologies that enable ad hoc wideband communication include, for example, IEEE 802.11b/g. Benefiting from the power capacities of vehicles, we assume that each MH has a significant transmission range and virtually unlimited power lifetime [5]. The architecture also supports handheld mobile devices.

In Fig. 4, when an MH p issues a spatial query, it tunes in to the broadcast channel and waits for the data. In the meantime, p can collect cached spatial data from peers to harvest existing results in order to complete its own spatial query. Because memory space is scarce in mobile devices, we assume that each MH p caches a set of POIs in an MBR related to its current location. Since the POIs located inside the MBR were obtained from the wireless information server, we define the area bounded by the MBR as a *verified region* $p.VR$ with regard to p's location.

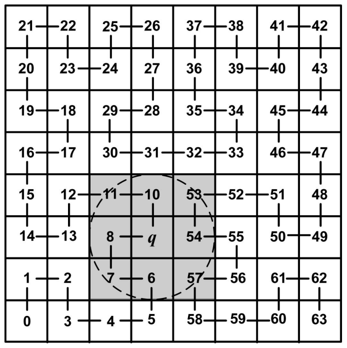


Fig. 5. An on-air kNN query example. The numbers represent index values.

3.3 Sharing-Based Nearest Neighbor Queries

Fig. 5 shows an example of an on-air kNN query based on the Hilbert curve index structure [31]. At first, by scanning the on-air index, the k-nearest object to the query point is found, and a minimal circle centered at q and containing all those k objects is constructed. The MBR of that circle, enclosing at least k objects, serves as the search range. Consequently, q has to receive the data packets that covers the MBR from the broadcast channel for retrieving its k-nearest objects. As shown in Fig. 5, the related packets span a long segment in the index sequence, that is, between 5 and 58, which will require a long retrieval time. The other problem with this search algorithm is that the indexing information has to be replicated in the broadcast cycle to enable twice scanning. The first scan is for deciding the kNN search range, and the second scan is for retrieving k objects based on the search range [31].

Therefore, we propose the *Sharing-Based Nearest Neighbor* (SBNN) query approach to improve the preceding on-air kNN query algorithm. The SBNN algorithm attempts to verify the validity of k objects by processing results obtained from several peers. Table 1 summarizes the symbolic notations used throughout this section.

3.3.1 Nearest Neighbor Verification

When an MH q executes SBNN, it first broadcasts a request to all its single-hop peers for their cached spatial data. Each peer that receives the request returns the verified region MBR and the cached POIs to q. Then, q combines the verified regions of all the replying peers, each bounded by

TABLE 1
Symbolic Notations

Symbol	Meaning
q	A query mobile host
$\mathbb{P}$	The set of all the peers that respond the query issued by q
$p.\mathbb{O}$	The cached POI set of a mobile host p where $p \in \mathbb{P}$
$p.VR$	The verified region of a mobile host p
M_{VR}	The merged verified region
e_s	The edge of M_{VR} which has the shortest distance to q
o_i	A nearest neighbor element in $p.\mathbb{O}$
H	A heap for storing SBNN query results. Its verified and unverified elements are defined as H.verified and H.unverified, respectively.
$\mathbb{O}$	The set of all the received POIs from peers
$\lvert\mathbb{A}\rvert$	The number of elements in set $\mathbb{A}$
$\lvert\lvert a, b\rvert\rvert$	The Euclidean distance between objects a and b

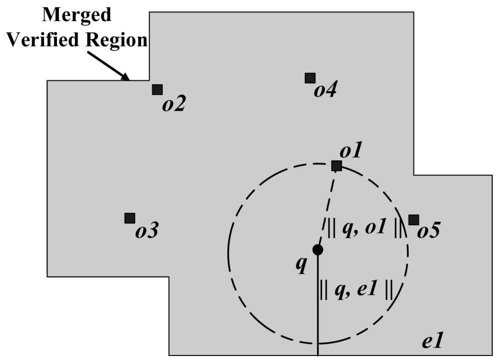


Fig. 6. Because e_1 has the shortest distance to q and $\|q, o_1\| \leq \|q, e_1\|$, POI o_1 is verified as a valid NN of MH q.

its MBR, into a merged verified region M_{VR} (the polygon in Fig. 6). The merging process is carried out by the *MapOverlay* algorithm [8] (line 4 in Algorithm 1). The core of SBNN is the *NN verification* (NNV) method, whose objective is to verify whether a POI o_i obtained from peers is a valid (that is, the top-k) NN of the MH q.

Let $\mathbb{P}$ denote the data collected by q from j peers $p_1, \cdots, p_j$. Consequently, the merged verified region M_{VR} can be represented as

$$M_{VR} = p_1.VR \cup p_2.VR \cup \cdots \cup p_j.VR.$$

Suppose that the boundary of M_{VR} consists of k edges, $\mathbb{E} = \{e_1, e_2, \ldots, e_k\}$, and there are l POIs, $\mathbb{O} = \{o_1, o_2, \ldots, o_l\}$, inside the M_{VR}. Let $e_s \in \mathbb{E}$ be the edge that has the shortest distance to q. An example is given in Fig. 6, where $k = 10$, and e_1 has the shortest distance to q.

Lemma 3.1. *Let $\widehat{\mathbb{O}} = \{\widehat{o}_1, \widehat{o}_2, \ldots, \widehat{o}_v\}$ be a set of POIs, where each is closer to q than e_s, and q is inside M_{VR}. Then, $\widehat{o}_1, \widehat{o}_2, \ldots, \widehat{o}_v$ are the* top-v *NNs of q.*

Proof. Assume that o_m is one of the top-v NNs of q but $o_m \notin \widehat{\mathbb{O}}$. Then, $\|q, o_m\| < \|q, \widehat{o}_v\|$, and $\|q, o_m\| < \|q, e_s\|$. Since $\|q, o_m\| < \|q, e_s\|$, o_m must be inside M_{VR}, and $o_m \in \mathbb{O}$. Based on the definition of $\widehat{\mathbb{O}}$, o_m must be a member of $\widehat{\mathbb{O}}$. However, this contradicts the assumption that $o_m \notin \widehat{\mathbb{O}}$. Therefore, $\widehat{\mathbb{O}}$ must cover the top-v NNs of q. □

In Fig. 6, according to Lemma 3.1, POI o_1 can be verified as the NN of q and is termed a *verified NN*, because the euclidean distance between o_1 and q is not greater than the euclidean distance between e_1 and q. Fig. 7 demonstrates a counter example. Since we are not sure if there is any POI within the unverified regions, o_4 cannot be verified as the top kNN of q. Note that there could be unverified regions inside the merged verified region.

The NNV method uses a heap H to maintain the entries of verified and unverified POIs discovered so far (see Table 2). Initially, H is empty. The NNV method inserts POIs to H as it verifies objects from MHs in the vicinity of q. The heap H maintains the POIs in ascending order in terms of their euclidean distances to q. Unverified objects are kept in H only if the number of verified objects is lower than what was requested by the query. The NNV method is formalized in Algorithm 1. Since the verified-region merging process dominates the algorithm complexity, the NNV method can be computed in $O(n \log n + i \log n)$ time,

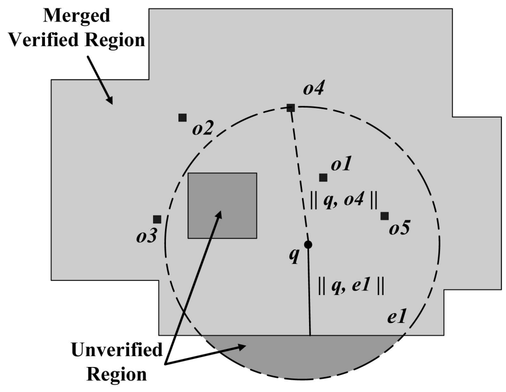


Fig. 7. Because of some unverified regions, o_4 cannot be verified as the top kNN of q.

where n is the total edge number of the two merged polygons, and i is the number of intersection points.

Algorithm 1: NNV (q, H, k)
```
 1: $P \leftarrow$ peer nodes responding to the query request
    issued from $q$.
 2: $M_{VR} \leftarrow \emptyset$
 3: for $\forall p \in \mathbb{P}$ do
 4:   $M_{VR} \cup = p.VR$ and $\mathbb{O} \cup = p.\mathbb{O}$
 5: end for
 6: $\forall o_i \in \mathbb{O}$, sort according to $\|q, o_i\|$
 7: Compute $\|q, e_s\|$, where edge $e_s$ has the shortest
    distance to $q$ among all the edges of $M_{VR}$
 8: $i = 1$
 9: while $|H| < k$ and $i \leq |\mathbb{O}|$ do
10:   if $\|q, o_i\| \leq \|q, e_s\|$ then
11:     $H$.verified $\cup = o_i$
12:   else
13:     $H$.unverified $\cup = o_i$
14:     $i{+}{+}$
15:   end if
16: end while
17: return $H$
```

If k elements in H are all verified by NNV, the kNN query is fulfilled. There will be cases when the NNV method cannot fulfill a kNN query. Hence, a set that contains unverified elements is returned. If the response time is critical, a user may agree to accept a kNN data set with unverified elements, where the objects are not guaranteed to be the top kNNs. However, the *correctness* of these approximate results can be estimated and will be discussed in the next section. If the result quality is the most important concern, the client has to wait until it receives all the required data packets from the broadcast channel. Nevertheless, the partial results in H can

TABLE 2
The Data Structure of the Heap H

POI	verified?	distance to q [miles]	correctness probability	surpassing distance $(r' - r)$
o_1	yes	2	100%	-
o_5	yes	3	100%	-
o_4	no	5	55%	2
o_3	no	6	40%	3

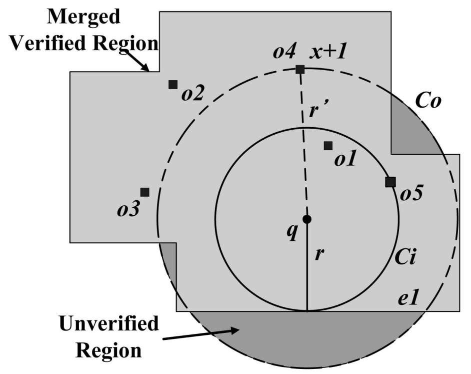


Fig. 8. The correctness probability of the unverified POI o_4 can be estimated based on the size of its unverified region.

be used to decrease the required data packets and thus speed up the on-air data collection (more details on this are in Section 3.3.3).

3.3.2 Approximate Nearest Neighbor

We calculate the probability that the unverified ith NN o of a query point q is actually the true ith NN of q. The reason that o cannot be verified is because there is a region that is not covered by q's neighboring peers. As long as a POI exists in the region, o cannot be q's ith NN. We denote such a region as o's unverified region. Fig. 8 shows an example. POI o_4 is the unverified third NN of q, because there is a possibility that another POI may exist in the shaded unverified region.

We assume that the POIs are *Poisson* distributed in our environment based on our experiments of several common POI types (gas stations, grocery stores, etc.) with chi-square tests [20], [24]. The probability of finding another POI in the unverified region U_i of an unverified POI o_i can be calculated with respect to the area of U_i. We formulate the correctness of an unverified POI based on the probability model in Lemma 3.2.

Lemma 3.2. *Assume that the POIs in an area E are Poisson distributed. Let q be a query MH that has retrieved x verified and y unverified NNs from M_{VR} for a kNN query. If the unverified region U_j of an unverified POI o_j of q covers the area of u square units, then the probability that o_j is the jth NN of q is $e^{-\lambda u}$, where λ denotes the average number of POIs per square unit.*

Proof. Let $\|q, o_j\| = r'$ and the circle C be defined by center point q with radius r'. According to the definition of the *Poisson* distribution, we have

$$P\{N(t+s) - N(s) = n\} = e^{-\lambda t}\frac{(\lambda t)^n}{n!}, \quad n = 0, 1, \ldots. \tag{1}$$

With the memoryless property of the *Poisson* distribution, we map t to the unverified region U_j within C and s to the verified region within C. $N(t)$ represents the total number of POIs that are located inside U_j. Since we know that the area of the unverified region of o_j is u square units, the probability that there is no POI in u square units is $e^{-\lambda u}$. □

Fig. 8 shows an example. Suppose that we obtain the average number of POIs per square unit as 0.3 (the value of λ) and the unverified region of o_4 covers two square units. We can then calculate the accuracy ratio of o_4 as the true third nearest POI of q as $e^{-0.6} \approx 0.5488$. Therefore, the probability that o_4 is the true third nearest POI of q is 55 percent.

In addition, the distance relationship between the last verified POI o_{lv} and an unverified POI o_u is also a useful metric, as demonstrated in Fig. 8. We name the metric the *surpassing distance* of the different between $\|q, o_u\|$ and $\|q, o_{lv}\|$ based on the euclidean distance. For example, if a motorist decides to take o_4 in the heap H (see Table 2) as his destination, in the worst case (o_4 is not the true third NN, and the true third NN is a little bit farther than o_5), he has to drive approximately 2 miles more.

The correctness probability and the surpassing distance of these unverified POIs are also memorized in the heap H, and they can be utilized by applications with different result quality requirements.

3.3.3 Broadcast Channel Data Filtering

Under most conditions, there are verified and unverified entries in H when the NNV method cannot totally fulfill a kNN query. For applications that require accurate NN information, we can utilize the partial results to calculate *data packet search bounds* from the entries in heap H to speed up the on-air NN search process. The heap H is in one of the six different states after an MH has executed the NNV mechanism without retrieving k verified objects:

- *State 1*. H is full and contains both verified and unverified entries.
- *State 2*. H is full and contains only unverified entries.
- *State 3*. H is not full and contains both verified and unverified entries.
- *State 4*. H is not full and contains only verified entries.
- *State 5*. H is not full and contains only unverified entries.
- *State 6*. H contains no entries.

In state 1, there may exist some POIs that are closer to q compared with the last element in H. Hence, we can consider the last entry of H as the final candidate NN in the NN search and utilize its distance as the search upper bound. In addition, the distance attribute d_v of the last verified entry can be another bound, that is, the search lower bound. Since we are certain about the POIs within the circle region C_i with radius d_v and center point q, q does not have to receive any data packet that contains objects completely covered by C_i. Conversely, when H is full and contains just unverified entries, we can infer only the upper bound (state 2). In states 3 and 4, after the MH performed the NNV algorithm, there have been merely less than k POIs found. Therefore, we can only infer the lower bound from the distance attribute of the last verified element in H. In the last two states, H is not full and contains only unverified entries or no entry at all. Consequently, we cannot infer any search bounds from them.

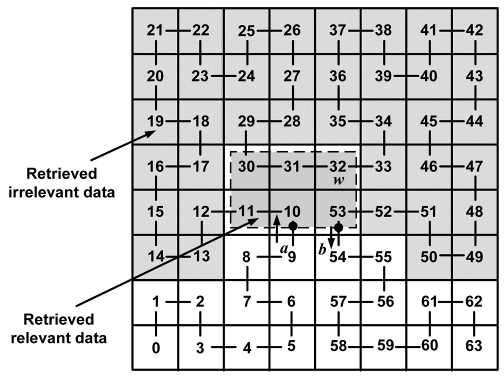


Fig. 9. A WQ on the Hilbert-curve index structure.

Based on the discussion in Sections 3.3.1, 3.3.2, and 3.3.3, the complete procedure of SBNN is presented in the following:

Algorithm 2: SBNN (q, H, k)

```
1: $H = \mathrm{NNV}(q, H, k)$
2: **if** ($|H.verified| = k$) or ($|H| = k$ and $accept = $ true) **then**
3:     **return** $H$
       {if $k$ verified NNs have been retrieved or the heap is full and $q$ accepts approximate results.}
4: **end if**
   {if $H$ is not full or $q$ disallows any approximate results, utilize the search upper and lower bounds to improve the on-air query efficiency.}
5: $H \cup k$NN query results returned from the updated on-air NN query.
6: **return** $H$
```

3.4 Sharing-Based Window Queries

As proposed in [31], the basic idea for an MH to process a WQ w based on a space-filling curve index is to decide a candidate set of points along the curve. The candidate set includes all the points that fall within the query window of w. Then, the MH retrieves the related packets and filters out data objects that are located outside the query window. As illustrated in Fig. 9, the dashed-line rectangle represents the query window of w. We can find the first point a and the last point b according to the order in which they occur on the Hilbert curve. Consequently, all the points inside this query window must lie on the Hilbert curve segmented by points a and b.

Although the algorithm proposed in [31] can find entry and exit bounding points on the Hilbert curve index to decrease the number of candidate points, the *access latency* is still very long. As shown in the example, the required data packets span between index values 9 and 54 and cover around 70 percent of the whole data file (the shaded area in Fig. 9). Although a search space partition technique was proposed in [31] to improve the performance, it still cannot mitigate the overhead of access latency. Therefore, we propose the *Sharing-Based Window Query* (SBWQ) method to improve the current on-air WQ algorithm.

For SBWQ, an MH q has to merge peer-verified regions $p.VR$ and collect related POI data from peers. Then, q computes the spatial relationship between the query window of w and the *merged verified region* M_{VR}. If w can be totally covered by M_{VR}, the WQ can be fulfilled. Otherwise, the whole or part of the query window must be solved as an on-air WQ. However, under the latter conditions, we may be able to reduce the query window.

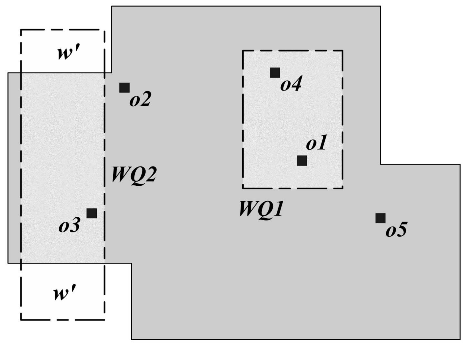


Fig. 10. POIs o_1 and o_4 are the query results of SBWQ $WQ1$.

3.4.1 Window Query Verification

The MH q first broadcasts a request to all its single-hop peers for requesting their cached spatial data. Then, it combines the returned verified regions $p.VR$, each bounded by its MBR, into a merged verified region M_{VR}. Next, q computes the spatial relationship between the query window w and M_{VR}. If w falls entirely inside M_{VR}, SBWQ will return the POIs that overlap with w (for example, $WQ1$ in Fig. 10).

3.4.2 Broadcast Channel Data Filtering

There will be cases when the SBWQ algorithm can provide only a partial result to a WQ (for example, $WQ2$ in Fig. 10). Consequently, one (or several) updated (that is, reduced) query window(s) w' will be utilized to decide the new search bound on the Hilbert curve index. Hence, the on-air WQ algorithm is executed for solving w'. Since w' is much smaller than w in many cases, the access latency can be markedly decreased. The SBWQ algorithm is formalized as follows:

Algorithm 3: SBWQ(q, w)

```
1: $P \leftarrow$ peer nodes responding to the query request issued from $q$.
2: **for** $\forall p \in P$ **do**
3:     $M_{VR} \cup = p.VR$ and $O \cup = p.O$
4: **end for**
5: $WQ \leftarrow \forall o \in O$ that overlap with $w$
6: **if** $w \subset M_{VR}$ **then**
7:     **return** $WQ$
8: **else**
9:     $WQ \cup$ query results returned from the on-air window query with $w'$.
       {if $w \not\subset M_{VR}$, utilize $w'$ to compute the new search bounds and results.}
10:    **return** $WQ$
11: **end if**
```

3.5 The Relationship between the Verified Region Size and Query Window Size

Since the efficiency of our techniques is mainly based on the cached previous query results, we are interested in the relationship between the verified region size and the query

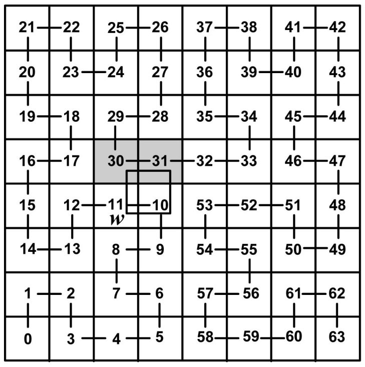


Fig. 11. The access latency of the WQ can be largely decreased by the cached data.

window size. We defined a metric, that is, the *access time saving ratio* (ATSR), for evaluating the relationship between the verified region size and the query window size. The ATSR is calculated by comparing the access latency with a certain verified area in cache versus the access latency without any verified region using the same query window size. Fig. 11 demonstrates an example. The merged verified region of an MH q covers broadcast cells 30 and 31, and the query window w overlaps with cells 10, 11, 30, and 31. Assume that the broadcast starts from cell 0. The MH q has to wait until the communication channel finishes the broadcasting of cell 31 before it can answer the query. However, with the aid of the verified region, q only needs to wait until the end of cell 11's transmission. Consequently, the MH can save 62.5 percent ($\frac{32-12}{32}$) of the access latency in this example. Note that the verified region size represents only around 3.1 percent ($\frac{2}{64}$) of the whole search space and the cached data is collected from numerous neighboring peers. In our experiments, we explore how much data an MH has to collect to achieve an ideal (maximum savings, given a specific cache size) ATSR.

Fig. 12 illustrates the relationship between the verified region size and the query window size, with the average values of 10,000 experiments. In Fig. 12a, we increased the verified region size from 1 percent to 20 percent of the whole search space, with a fixed query window size (2 percent of the whole search space) and the ATSR increasing from 3 percent to 70 percent. As demonstrated in the figure, if the verified region is around 5 percent of the whole search space, we can save more than 50 percent of access latency. In addition, we also enlarged the query window size from 1 percent to 20 percent, with a constant verified region size (6 percent) and the saved access latency becoming very limited when the query window size is larger than 10 percent, as shown in Fig. 12b. However, we usually have relatively small WQs in most location-based service applications [27].

4 Simulation Performance Evaluation

To evaluate the performance of our approach, we have implemented the sharing-based spatial query algorithms within a simulator. In addition to enabling efficient and decentralized applications, the objective of our P2P design is to decrease access latency in two dimensions. First, the access latency can be reduced as queries are answered directly by peers. Second, for the remaining queries that require packets from the broadcast channel, our technique diminishes the required number of packets by providing search bounds for the spatial query algorithms. Consequently, the focus of our simulations is to quantify the access latency variations as a function of two main parameters: the *Peer Query-Fulfilling Rate* (PQFR) and *Broadcast Packet Access Rate* (BPAR). PQFR quantifies what percentage of the client spatial query requests are fulfilled by peers, and BPAR denotes how many broadcast data packet are required compared with the solution in [31] for a sequence of queries with partial results from sharing-based queries. Our experiments were performed with both synthetic and real-world parameter sets.

4.1 Simulator Implementation

Our simulator consists of two main modules: the *MH module* and the *base station module*. The MH module generates and controls the movements and query launch patterns of all MHs. Each MH is an independent object that decides its movement autonomously. The base station module operates a broadcast channel for continuously sending data packets to MHs. Spatial data indexing is provided with the well-known Hilbert curve [16]. We implemented our SBNN and SBWQ algorithms in the MH module.

Each MH is implemented as an independent object that encapsulates all its related parameters such as the movement velocity $M_{Velocity}$, the cache capacity C_{Size}, and the

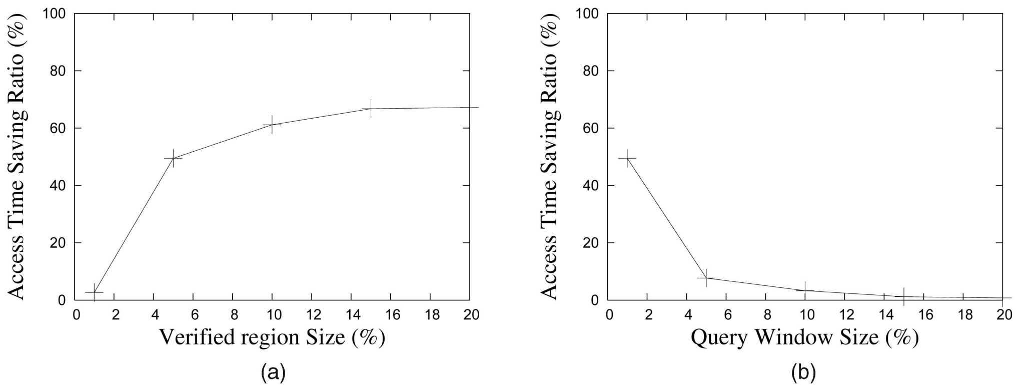


Fig. 12. The analytical results of various verified region sizes and query window sizes. (a) Verified region size. (b) Query window size.

TABLE 3
Parameters for the Simulation Environment

Parameter	Description
POI_{Number}	The number of point of interest in the system
MH_{Number}	The number of mobile hosts in the simulation area
C_{Size}	The cache capacity per data type of each mobile host
λ_{Query}	The mean number of queries per minute
Tx_{Range}	The wireless transmission range of a mobile host
λ_{kNN}	The mean number of queried nearest neighbors
λ_{Window}	The mean size of query windows
$\lambda_{Distance}$	The mean distance between a query MH and the center point of its query window
$T_{execution}$	The length of a simulation run

wireless transmission range Tx_{Range}. All MHs move inside a geographical area, measuring 15 miles × 15 miles. Additionally, user-adjustable parameters are provided for the simulation, such as execution length, the number of MHs and their query frequency, and the number of POIs. Table 3 lists all of the simulation parameters.

The simulation is initialized by randomly choosing a starting location for each MH within the simulation area. The movement generator then produces trajectories with an underlying road network. We employed the *random waypoint* model [4] as our mobility model. Each MH selects a random destination point inside the simulation area and progresses toward it. Upon reaching that location, it pauses for a random interval and decides on a new destination for the next travel period. This process repeats for all MHs until the end of the simulation.

Every simulation has numerous intervals (whose lengths are Poisson distributed), and during each interval, the simulator selects a random subset of MHs to launch spatial queries (the query intervals are also based on the Poisson distribution). The subset size is controlled via the λ_{Query} parameter (for example, 1,000 queries per minute). These MHs then execute the SBNN or SBWQ algorithms by interacting with their peers. An MH will first attempt to answer each spatial query via the sharing-based approach. If this is unsuccessful, the query will be solved by listening to the broadcast channel. Each MH manages its local query result cache with a combination of the following policies:

1. An MH stores all the verified POIs and their minimum bounding boxes. The cache replacement policy is based on the current moving direction and the data distance between the current location of the MH and the location of a data object [23].
2. If a spatial query must be solved by listening to the broadcast channel, the MH will store as many received POIs as its cache capacity allows (for example, for a 5-NN query, if the downloaded broadcast packets contain 15 POIs and the cache capacity is 30 POIs for each data type, the MH will store all of them and their collective MBR).

The SBNN query algorithm is implemented according to the method detailed in Section 3.3. Multiple potentially overlapping MBRs must be combined to provide the verified region. The simulator sequentially merges peer-returned MBRs into a *merged verified region* M_{VR} by performing the *MapOverlay* algorithm and also combines the returned POIs into a candidate list O. Afterward, an MH sequentially verifies the objects in O with our verification technique based on M_{VR}. Similarly, we implemented the SBWQ algorithm (Section 3.4) in the simulator.

4.1.1 Simulation Parameter Sets

To obtain results that closely correspond to real-world conditions, we obtained our simulation parameters from public data sets, for example, car and gas station densities in urban areas. We term the two parameter sets based on these real-world statistics the *Los Angeles* parameter set and the *Riverside County* parameter set.

- POIs. We obtained information about the density of the of-interest objects (for example, gas stations, restaurants, and hospitals) in Southern California from two online sites: GasPriceWatch.com[2] and CNN/Money. Because gas stations are commonly the target of spatial queries, we use them as the sample POI types for our simulations. The PQFR of other POI types are expected to be very similar.
- MHs. We collected the vehicle statistics of Southern California from the Federal Statistics Web site. The data provide the number of registered vehicles in Los Angeles and Riverside County (1,092,939 and 944,645, respectively). In our simulations, we assume that about 10 percent of these vehicles are on the road during nonpeak hours according to the traffic information from Caltrans.[3] We further obtained the land area of each region to compute the average vehicle density per square mile.

The Los Angeles and Riverside County parameter sets represent a very dense urban area and a low-density more rural area. Hence, for comparison purposes, we blended the two real parameter sets to generate the third synthetic set. The synthetic data set demonstrates vehicle and interest object densities in between Los Angeles and Riverside County, representing a suburban area. Table 4 lists the three parameter sets.

4.2 Performance of the kNN Query

We utilized all three simulation parameter sets to evaluate our peer sharing techniques for solving kNN queries. We varied the following parameters to observe their effects on the system performance: the wireless transmission range, the cache capacity, and the NN number k. The performance metric in the MH module was PQFR. The key differences between the three parameter sets are their vehicle and POI densities. Hence, we utilized the simulation to evaluate the applicability of our design to different geographical areas. All simulation results were recorded after the system model reached a steady state.

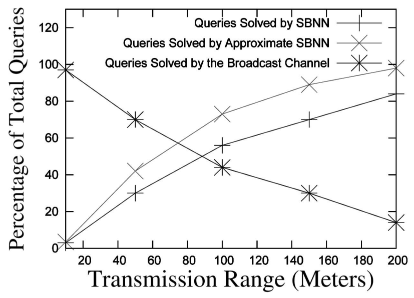
4.2.1 Transmission Range Experiments

We first varied the MH wireless transmission range from 10 to 200 m, with all the other parameters unchanged. Although the reliable coverage range for IEEE 802.11b/g in open space with good antennas can be more than 300 m [9], obstacles such as buildings could diminish the range to 200 m or less in urban areas. Therefore, we chose 200 m as a

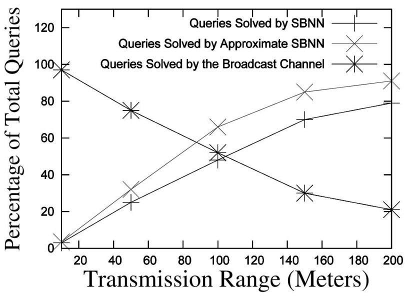
2. http://www.gaspricewatch.com.
3. http://www.dot.ca.gov/hq/traffops/saferesr/trafdata/.

TABLE 4
The Simulation Parameter Sets

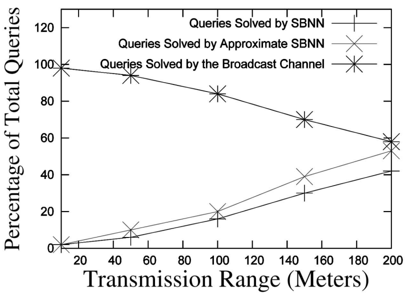
Parameter	Los Angeles City	Riverside County	Synthetic Suburbia	Units
POI_{Number}	2750	1450	2100	
MH_{Number}	93300	9700	51500	
C_{Size}	50	50	50	
λ_{Query}	6220	650	3440	$\min^{-1}$
Tx_{Range}	200	200	200	m
λ_{kNN}	5	5	5	
λ_{window}	3	3	3	%
$\lambda_{Distance}$	1	1	1	mile
$T_{execution}$	10	10	10	hr



(a)

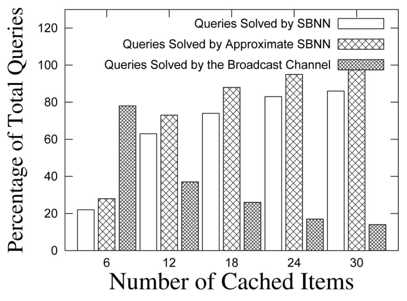


(b)

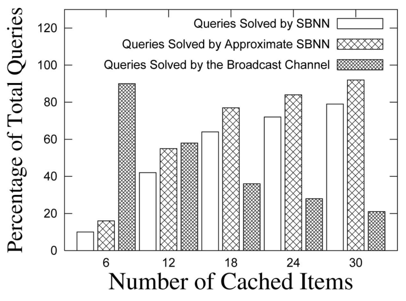


(c)

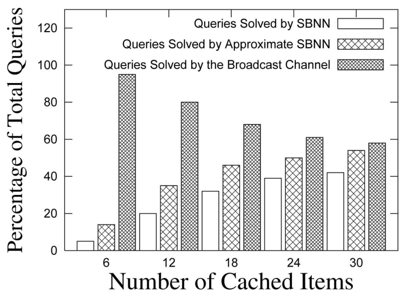
Fig. 13. The percentage of resolved queries as a function of the wireless transmission range. (a) Los Angeles. (b) Synthetic suburbia. (c) Riverside County.



(a)



(b)



(c)

Fig. 14. The percentage of resolved queries as a function of the MH cache capacity. (a) Los Angeles. (b) Synthetic suburbia. (c) Riverside County.

practical transmission upper limit. Fig. 13 demonstrates the percentage of queries that can be resolved by SBNN, approximate SBNN (with the POI correctness probability higher than 50 percent), or the broadcast channel with the three experimental parameter sets. As the transmission range extends, an increasing number of queries can be answered by surrounding peers. Because of its high vehicle density, the effect is most prominent with the Los Angeles parameter set. With a 200-m transmission range, less than 20 percent of the queries must be solved by listening to the broadcast channel for exact results.

4.2.2 Cache Capacity Experiments

Next, we tested the impact of various MH cache capacities, which denote how many POI objects an MH can store. Fig. 14 illustrates the increase in the cache capacity from 6 to 30 with the three parameter sets. Even though the total number of of-interest objects is much larger than the maximum cache capacity, we observe a remarkable increase in queries solved by SBNN with a higher MH cache capacity in Figs. 14a and 14b.

4.2.3 Nearest Neighbor Number k Experiments

To see the effect of varying the number of requested NNs, that is, k, we altered k in the range from 3 to 15 as the mean number for each query. As shown in Fig. 15, the solved queries by the broadcast channel for the Los Angeles parameter set increased by 28 percent when we raised k from 3 to 15. The solved queries for the Riverside County parameter set increased by only 21 percent, because its starting level was much higher. Not surprisingly, our technique is much more effective for small values of k.

4.3 Performance of Window Queries

Similar to Section 4.2, we utilized all three experimental parameter sets to evaluate our peer sharing techniques for solving WQs. We varied three parameters, that is, the wireless transmission range, the cache capacity, and the query window size, to observe their influence on the system performance.

4.3.1 Transmission Range Experiments

In this experiment, we varied the MH wireless transmission range from 10 to 200 m, with all the other parameters

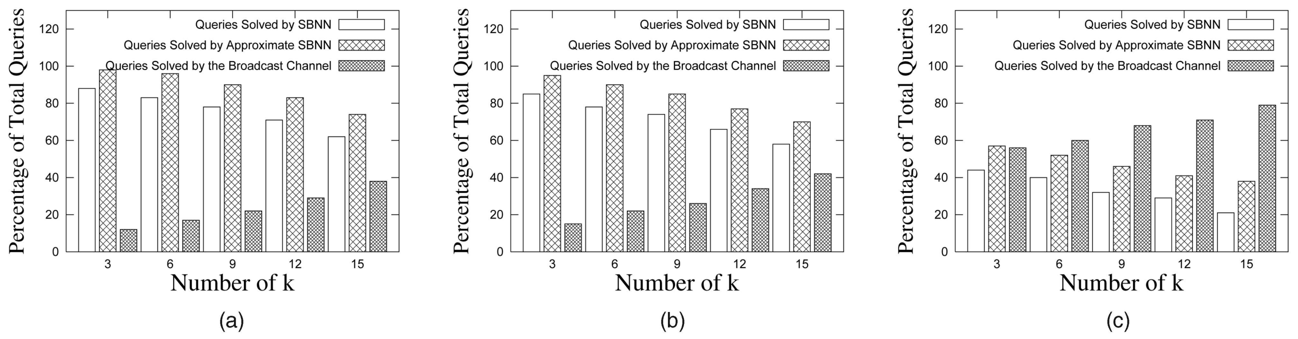


Fig. 15. The percentage of resolved queries as a function of k. (a) Los Angeles. (b) Synthetic suburbia. (c) Riverside County.

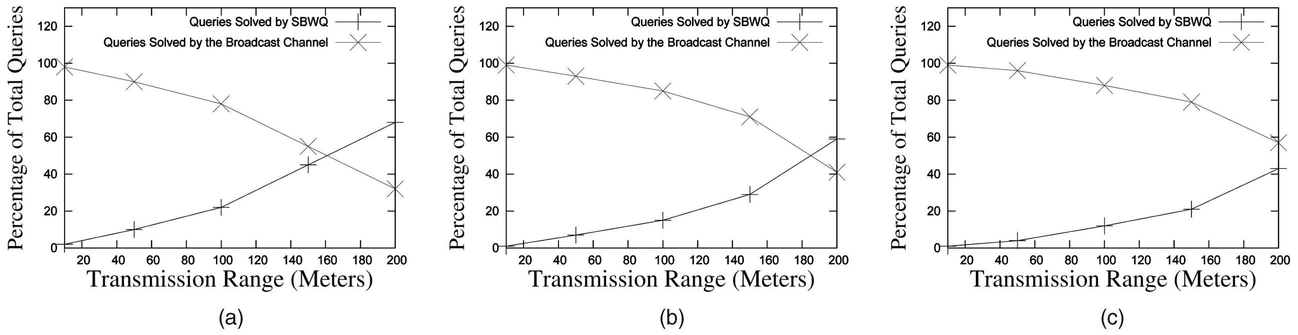


Fig. 16. The percentage of resolved queries as a function of the wireless transmission range. (a) Los Angeles. (b) Synthetic suburbia. (c) Riverside County.

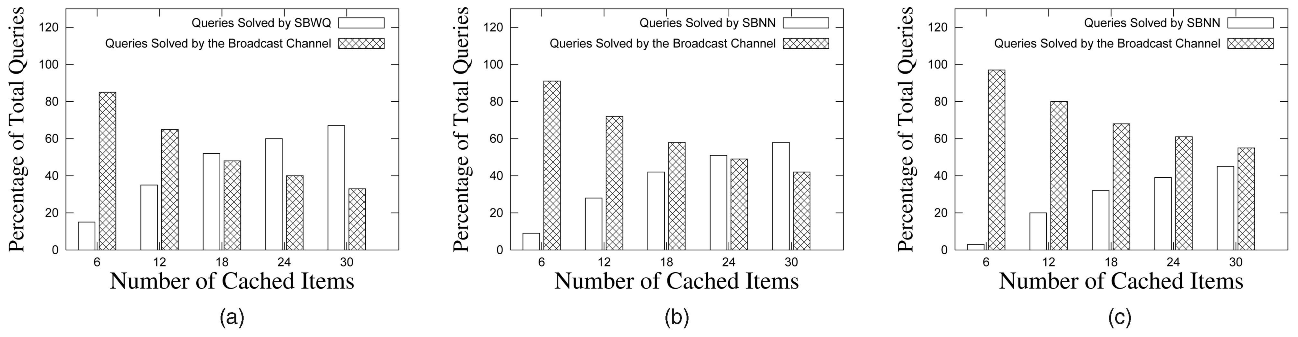


Fig. 17. The percentage of resolved queries as a function of the MH cache capacity. (a) Los Angeles. (b) Synthetic suburbia. (c) Riverside County.

unchanged. Fig. 16 demonstrates the proportion of WQs that can be resolved by SBWQ or the broadcast channel with the three parameter sets. The trend of the simulation results is similar to the kNN case. With increasing transmission range, more queries can be fulfilled by surrounding peers.

4.3.2 Cache Capacity Experiments

We studied the effect of various MH cache capacities by enlarging the cache capacity from 6 to 30 with the three parameter sets, and the results are shown in Fig. 17. We observed that with the increase in cache capacity, more WQs can be fulfilled by peers. Therefore, MHs can have shorter access latencies with higher cache capacities.

4.3.3 Query Window Size Experiments

We examined the effect that varying the query window size would have on the system performance. In our experiment, we varied the query window size from 1 percent to 5 percent of the whole search space. The center location of the query window is randomly chosen with a distance to the query MH based on a normal distribution. Fig. 18 illustrates the results. With a relatively small query window (less than 3 percent), over 50 percent of the WQs can be fulfilled through our sharing mechanism.

From all the performed experiments, we observed that the MH density has a considerable impact on system performance. Consequently, if more MHs travel in a specific area, each MH has a higher opportunity to fulfill its spatial queries by peers and hence decrease the access latency.

4.4 Experimental Results of the Broadcast Packet Access Rate

In order to evaluate the spatial query search bounds in Sections 3.3 and 3.4, we extended the OASQ algorithms proposed in [31] with search bounds. The performance metric for comparing the extended OASQ (EOASQ) and OASQ is *BPAR*. For each spatial query that cannot be fulfilled by our sharing-based mechanism, the MH module executes both OASQ and EOASQ algorithms to compare the performance improvement with respect to the packet

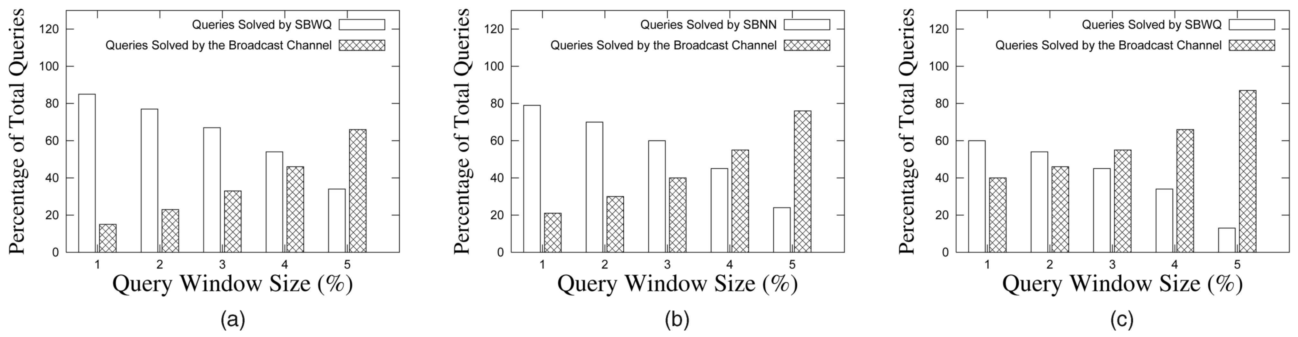


Fig. 18. The percentage of resolved queries as a function of query window size. (a) Los Angeles. (b) Synthetic suburbia. (c) Riverside County.

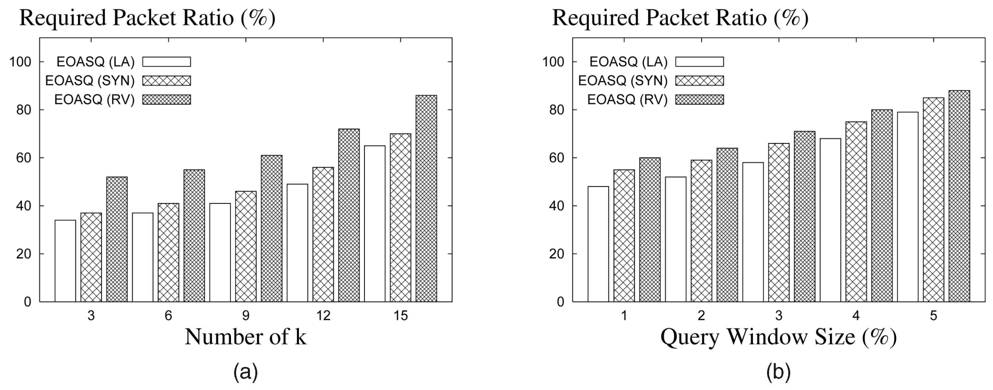


Fig. 19. The packet access comparison between EOASQ and OASQ. We normalized the required packet number of EOASQ to OASQ. (a) kNN queries. (b) WQs.

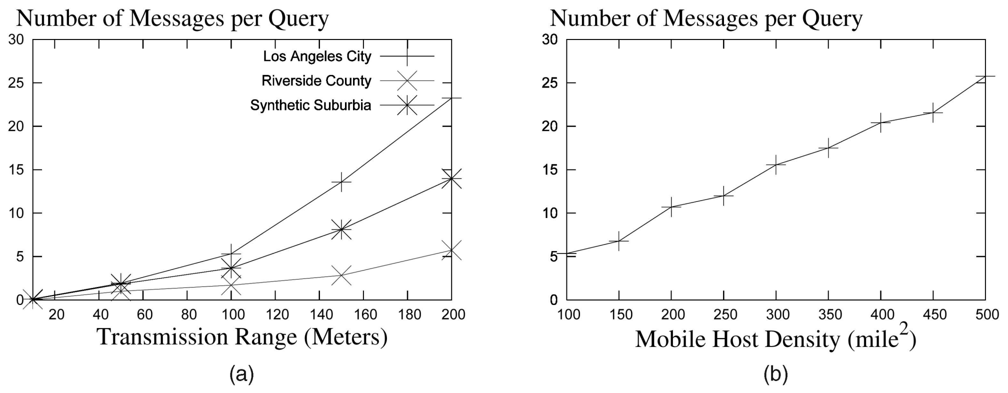


Fig. 20. The energy cost analysis of the proposed approach. (a) Transmission range increment. (b) MH density increment.

access of the broadcast channel. We examined the behavior of the original and our extended solutions as the number of k and query window size increase. Because spatial queries are generated by randomly selected MHs, query points are uniformly distributed over the simulation area.

Since EOASQ usually requests fewer data packets than OASQ, we believe that our search bounds can decrease the access latency and tuning time. During the simulation process, the MH module counts the number of data packet accesses that correspond to both access latency and tuning time. As shown in Fig. 19, the EOASQ algorithm performs consistently better than OASQ with various numbers of k and query window sizes. We conclude that the search bound technique can effectively decrease the number of broadcast packet accesses. We varied the number of k from 3 to 15 with the three parameter sets, and the EOASQ algorithm accesses 66 percent to 14 percent fewer packets than OASQ. Similarly, the EOASQ accesses 51 percent to 12 percent fewer packets than OASQ when we increased the query window size from 1 percent to 5 percent.

We conclude from all the performed experiments that the MH density has a considerable impact on the *PQFR*. As a result, if more MHs travel in a specific area, each MH has a higher opportunity to fulfill its spatial queries by peers. Furthermore, the spatial query search bounds also have a significant positive effect on the *BPAR* and successfully decrease the access latency and tuning time.

4.5 Energy Cost Analysis of the Proposed Approach

Although in this research, we applied our approach to vehicles that have virtually unlimited power lifetime, we measured the number of message transmissions to analyze the energy related cost of our approach. As illustrated in Fig. 20a, we utilized the three parameter sets to observe the increase in communication messages between an MH and its peers when extending the wireless transmission range from 10 to 200 m. Since a mobile user always broadcasts data requests, the message counts represent the average total response messages from peers for a spatial query. Because the transmission range of an MH covers a 2D area, the message count grows quadratically with all the three parameter sets. Similarly, we can see a steady message count increase when the MH density per square mile is raised from 100 to 500 in Fig. 20b (with a fixed 200-m transmission range). Consequently, as illustrated through the experimental results, the energy cost for solving a query will expand when we extend the wireless transmission range. In addition, the mobile user density also has a significant influence on the energy consumption.

5 Conclusion

This paper presented a novel approach for reducing the spatial query access latency by leveraging results from nearby peers in wireless broadcast environments. Significantly, our scheme allows a mobile client to locally verify whether candidate objects received from peers are indeed part of its own spatial query result set. The experiment results indicate that our method can reduce the access to the wireless broadcast channel by a significant amount, for example, up to 80 percent, in a dense urban area. This is achieved with minimal caching at the peers. By virtue of its P2P architecture, the method exhibits great scalability: the higher the mobile peer density, the more the queries answered by peers. Therefore, the query access latency can be markedly decreased with the increase in clients.

Acknowledgments

This research has been funded in part by the US National Science Foundation (NSF) Grants EEC-9529152 (IMSC ERC), CMS-0219463 (ITR), and IIS-0534761 and by equipment gifts from the Intel Corp., Hewlett-Packard, Sun Microsystems, and Raptor Networks Technology.

References

[1] S. Acharya, R. Alonso, M.J. Franklin, and S.B. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communications Environments," *Proc. ACM SIGMOD '95,* pp. 199-210, 1995.

[2] D. Barbará, "Mobile Computing and Databases: A Survey," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, no. 1, pp. 108-117, Jan./Feb. 1999.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," *Proc. ACM SIGMOD '90,* pp. 322-331, 1990.

[4] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J.G. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proc. ACM MobiCom '98,* pp. 85-97, 1998.

[5] V. Bychkovsky, B. Hull, A.K. Miu, H. Balakrishnan, and S. Madden, "A Measurement Study of Vehicular Internet Access Using In Situ Wi-Fi Networks," *Proc. ACM MobiCom '06,* Sept. 2006.

[6] C.-Y. Chow, H. Va Leong, and A. Chan, "Peer-to-Peer Cooperative Caching in Mobile Environment," *Proc. 24th IEEE Int'l Conf. Distributed Computing Systems Workshops (ICDCSW '04),* pp. 528-533, 2004.

[7] C.-Y. Chow, H. Va Leong, and A.T.S. Chan, "Distributed Group-Based Cooperative Caching in a Mobile Broadcast Environment," *Mobile Data Management,* pp. 97-106, 2005.

[8] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry Algorithms and Applications,* second ed. Springer, 2000.

[9] G. Gaertner and V. Cahill, "Understanding Link Quality in 802.11 Mobile Ad Hoc Networks," *IEEE Internet Computing,* vol. 8, no. 1, pp. 55-60, 2004.

[10] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," *Proc. ACM SIGMOD '84,* pp. 47-57, June 1984.

[11] T. Hara, "Cooperative Caching by Mobile Clients in Push-Based Information Systems," *Proc. 11th ACM Int'l Conf. Information and Knowledge Management (CIKM '02),* pp. 186-193, 2002.

[12] T. Hara and S.K. Madria, "Data Replication for Improving Data Accessibility in Ad Hoc Networks," *IEEE Trans. Mobile Computing,* vol. 5, no. 11, pp. 1515-1532, Nov. 2006.

[13] G.R. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases," *ACM Trans. Database Systems,* vol. 24, no. 2, pp. 265-318, 1999.

[14] H. Hu, J. Xu, W. Sing Wong, B. Zheng, D. Lun Lee, and W.-C. Lee, "Proactive Caching for Spatial Queries in Mobile Environments," *Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE '05),* pp. 403-414, 2005.

[15] T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Data on Air: Organization and Access," *IEEE Trans. Knowledge and Data Eng.,* vol. 9, no. 3, pp. 353-372, May-June 1997.

[16] H.V. Jagadish, "Analysis of the Hilbert Curve for Representing Two-Dimensional Space," *Information Processing Letters,* vol. 62, no. 1, pp. 17-22, 1997.

[17] W.-S. Ku and R. Zimmermann, "Location-Based Spatial Queries with Data Sharing in Mobile Environments," *Proc. 22nd IEEE Int'l Conf. Data Eng. (ICDE '06) Workshops,* p. 140, 2006.

[18] W.-S. Ku, R. Zimmermann, and H. Wang, "Location-Based Spatial Queries with Data Sharing in Wireless Broadcast Environments," *Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE),* 2007.

[19] M.F. Mokbel, C.-Y. Chow, and W.G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," *Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB '06),* pp. 763-774, 2006.

[20] D.C. Montgomery and G.C. Runger, *Applied Statistics and Probability for Engineers,* second ed. John Wiley & Sons, 1998.

[21] W.-C. Peng and M.-S. Chen, "Design and Performance Studies of an Adaptive Cache Retrieval Scheme in a Mobile Computing Environment," *IEEE Trans. Mobile Computing,* vol. 4, no. 1, pp. 29-40, Jan./Feb. 2005.

[22] W.-C. Peng and M.-S. Chen, "Shared Data Allocation in a Mobile Computing System: Exploring Local and Global Optimization," *IEEE Trans. Parallel and Distributed Systems,* vol. 16, no. 4, pp. 374-384, Apr. 2005.

[23] Q. Ren and M.H. Dunham, "Using Semantic Caching to Manage Location-Dependent Data in Mobile Computing," *Proc. ACM MobiCom '00,* pp. 210-221, 2000.

[24] J.L. Ringuest, "A Chi-Square Statistic for Validating Simulation-Generated Responses," *Computers and Operations Research,* vol. 13, no. 4, pp. 379-385, 1986.

[25] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," *Proc. ACM SIGMOD '95,* pp. 71-79, 1995.

[26] T.K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," *Proc. 13th Int'l Conf. Very Large Data Bases (VLDB '87),* pp. 507-518, 1987.

[27] S. Spiekermann, "General Aspects of Location-Based Services," *Location-Based Services,* pp. 9-26, 2004.

[28] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *IEEE Trans. Mobile Computing,* vol. 5, no. 1, pp. 77-89, Jan. 2006.

[29] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. Lun Lee, "Location-Based Spatial Queries," *Proc. ACM SIGMOD '03,* pp. 443-454, 2003.

[30] B. Zheng and D. Lun Lee, "Information Dissemination via Wireless Broadcast," *Comm. ACM,* vol. 48, no. 5, pp. 105-110, 2005.

[31] B. Zheng, W.-C. Lee, and D. Lun Lee, "Spatial Queries in Wireless Broadcast Systems," *Wireless Networks,* vol. 10, no. 6, pp. 723-736, 2004.

Wei-Shinn Ku received the MS degree in computer science, the MS degree in electrical engineering, and the PhD degree in computer science from the University of Southern California (USC) in 2003, 2006, and 2007, respectively. He is currently an assistant professor in the Department of Computer Science and Software Engineering, Auburn University. His research interests include spatiotemporal data management, geographical information systems, network security, and peer-to-peer systems. He is a member of the IEEE and the ACM.

Roger Zimmermann is currently an associate professor in the Department of Computer Science, National University of Singapore. He is on the editorial board of the *ACM Computers in Entertainment Magazine*, the *ACM Transactions on Multimedia Computing, Communications and Applications*, and the *International Journal of Multimedia Tools and Applications*. He has served on the conference program committees of many leading conferences. His research interests are distributed and peer-to-peer systems, collaborative environments, streaming media architectures, geospatial database integration, and mobile location-based services. He is a coauthor of a book and more than 85 conference proceedings, journal articles, and book chapters in the areas of multimedia and databases. He is the holder of two patents. He is a member of the ACM and a senior member of the IEEE.

Haixun Wang received the BS and MS degrees in computer science from Shanghai Jiao Tong University in 1994 and 1996, respectively, and the PhD degree in computer science from the University of California, Los Angeles, in 2000. He is currently the technical assistant to the head of computer science at the IBM T.J. Watson Research Center. He has served on the program committees of international conferences and workshops and has been a reviewer for some leading academic journals in the database field. He has published more than 90 research papers in refereed international journals and conference proceedings. He is a member of the IEEE, the ACM, the ACM SIGMOD, and the ACM SIGKDD.