Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Monitoring continuous *k*-nearest neighbor queries in the hybrid wireless network*

Young-Mo KWON, HaRim JUNG, Yon Dohn CHUNG‡

(*Department of Computer Science and Engineering, Korea University, Seoul 136-713, Korea*)

E-mail: {gkgk0524, harim3826, ydchung}@korea.ac.kr

Received Apr. 1, 2010; Revision accepted Aug. 11, 2010; Crosschecked Jan. 31, 2011

Abstract: In a mobile/pervasive computing environment, one of the most important goals of monitoring continuous spatial queries is to reduce communication cost for location-updates. Existing work uses many cellular wireless connections, which would easily become the performance bottleneck of the overall system. This paper introduces a novel continuous *k*NN query monitoring method to reduce communication cost in the hybrid wireless network, where the moving objects in the wireless broadcasting system construct the ad-hoc network. Simulation results prove the efficiency of the proposed method, which leverages the wireless broadcasting channel as well as the WiFi link to alleviate the burden on the cellular uplink communication cost.

Key words: Continuous *k*NN query monitoring, Ad-hoc networks, Wireless broadcasting systems
doi:10.1631/jzus.C1000080 **Document code:** A **CLC number:** TP393

1 Introduction

With advances in wireless technologies and the widespread use of hand-held computing devices, the trend toward mobile/pervasive computing has gained momentum (Akyildiz *et al*., 2002). Since location (e.g., 2D coordinates) is one of the most important properties in a pervasive computing environment, various location based services (LBSs) have emerged as the promising applications (Lee *et al*., 2002; Junglas and Watson, 2008). Accordingly, a large number of mobile clients are expected to access a variety of valuable information from anywhere and at any time. According to eMarketer.com, the number of mobile phone based LBS subscribers will be 486 million by 2012.

Recently, monitoring continuous *k*-nearest neighbor (*k*NN) queries over moving objects attracts considerable attention for the support of LBSs (Hu *et al*., 2005; Mouratidis *et al*., 2005a; 2005b; Xiong *et al*., 2005; Yu *et al*., 2005). Given a collection of moving objects and a long-running continuous *k*NN query *q*, issued from a client, *k*NN monitoring continually computes and updates the closest *k* objects from a given query point *q.p* until *q* expires. A straightforward strategy, namely a centralized continuous *k*NN monitoring strategy (Mouratidis *et al*., 2005a; Xiong *et al*., 2005; Yu *et al*., 2005), is to have each moving object periodically report its current location. Then, the server keeps re-evaluating *q* and informing the client whenever the result of *q* changes. However, where the server involves a large number of moving objects and/or continuous *k*NN queries, the performance may degrade quickly due to the overwhelming workload as well as a severe communication bottleneck.

To remedy the problem mentioned above, some recent researches (Hu *et al*., 2005; Mouratidis *et al*., 2005b) have followed a distributed continuous *k*NN monitoring strategy, where each moving object (1) monitors a number of the associated queries and (2) sends its current location to the server only when the results of those queries are influenced by its movement. This leads moving objects to communicate with

‡ Corresponding author
* Project supported by the second stage of the Brain Korea 21 Project

the server much less frequently. Nevertheless, the server involves the frequent dissemination of the relevant information to support the monitoring tasks at the moving object side. As a result, a more scalable solution with regard to communication cost is required.

Wireless broadcasting (Imielinski *et al*., 1997; Lee *et al*., 2002) is considered to be an effective solution for the communications between the server and moving objects, since it accommodates an unlimited number of moving objects simultaneously at a constant cost. In this paper, we propose a new distributed continuous *k*NN monitoring method in the hybrid wireless network, where moving objects construct the wireless ad-hoc network in the wireless broadcasting system. All the moving objects in the system are assumed to be location-aware and have two radio interfaces, one for WiFi and the other for cellular networks. Using WiFi links, the moving objects send their current locations to some designated moving objects, which aggregate these locations and send the message to the server through the cellular uplink channels. Consequently, instead of having many cellular links as performed in the previous work mentioned above, our proposed method uses WiFi links together with a few cellular links to send the current locations of the moving objects to the server. In addition, the server sets aside a wireless broadcasting channel to periodically disseminate query information to the moving objects. The geographical area covered by the system is mapped onto a 2D regular grid of cells. Whenever a moving object moves from the current cell to another cell, it accesses the broadcasting channel to download the associated information for a query monitoring task without contacting the server.

2 Related work

Traditional research (Roussopoulos *et al*., 1995; Papadopoulos and Manolopoulos, 1997; Cheung and Fu, 1998; Hjaltason and Samet, 1999; Bohm, 2000) in spatial databases focused on processing the *k*NN query that finds the *k* closest data objects to a given static query point from a static dataset. They assumed that the data objects are indexed with spatial access methods, e.g., the R-tree (Guttman, 1984), and used some pruning heuristics to confine the search space.

Various methods have been proposed that deal with *k*NN query processing under dynamic environments, where objects/queries are constantly moving, (Tao and Papadias, 2002; Zhang *et al*., 2003). These methods commonly use the spatial and/or temporal validity information to answer *k*NN queries. However, they deal only with efficient processing of one-time snapshot *k*NN queries, and thus cannot support the queries running continuously over the database.

Motivated by LBSs, Mouratidis *et al*. (2005a), Xiong *et al*. (2005), and Yu *et al*. (2005) addressed the continuous *k*NN queries and focused on maintaining the up-to-date results of multiple long-running continuous *k*NN queries. They followed the centralized *k*NN monitoring strategy, and used the grid index structure. In addition, they did not assume the specific motion patterns of the moving objects. Specifically, in their approaches, the server (1) partitions the data space into a regular grid of cells to index moving objects, and (2) periodically re-evaluates the result of continuous *k*NN queries at each update cycle based on the location-updates received from the moving objects.

Yu *et al*. (2005) proposed two main-memory indexing methods with different scenarios for continuous *k*NN monitoring on moving objects, namely object indexing and query indexing. For the object indexing method, the overhaul query re-evaluation and the incremental query re-evaluation were introduced. Query indexing, which is suitable only for the small number of continuous *k*NN queries, indexes query points instead of moving objects. The authors also introduced the hierarchical version of the object indexing to improve the overall performance when the moving objects are not uniformly distributed.

Xiong *et al*. (2005) presented the shared execution algorithm for evaluating a large set of continuous *k*NN queries continuously (SEA-CNN). SEA-CNN uses an incremental *k*NN monitoring method, where the query re-evaluation algorithm considers only the moving objects that may affect the current results of continuous *k*NN queries. SEA-CNN assumes that the moving objects are indexed with a grid index in the secondary memory and focuses only on incrementally re-evaluating the query results without considering the initial query evaluation.

The best known method for continuous *k*NN query monitoring is the conceptual partitioning method (CPM) proposed by Mouratidis *et al*. (2005a).

CPM virtually organizes the cells of the grid into rectangles based on their proximity from each continuous *k*NN query point *q.p*. With the help of the virtual rectangles, the query re-evaluation algorithm visits the minimal set of cells.

Some recent research (Hu *et al*., 2005; Mouratidis *et al*., 2005b) followed the distributed computation strategy for monitoring continuous *k*NN queries, where the server pushes some monitoring tasks to the moving objects' side. This approach achieves significant savings in terms of server load as well as communication cost when compared with the centralized *k*NN monitoring strategy.

Hu *et al*. (2005) presented the first distributed monitoring method for continuous *k*NN queries, where they employed the concept of 'safe region'. Specifically, each moving object *o* was assigned a circular/rectangular area, such that *o* should report its current location to the server only when *o* exits from this area. Otherwise, *o* does not need to report its location since its movement does not affect the result of any continuous *k*NN query.

Mouratidis *et al*. (2005b) proposed a threshold-based distributed monitoring method. The basic idea is to categorize the moving objects into two sets for each continuous *k*NN query *q*: the inner objects and the outer objects. Specifically, the objects that currently belong to the result of *q* are called the inner objects and the remaining objects are called the outer objects. In addition, for each object *o*, a threshold was calculated so that as long as *o* is within the range defined by the threshold, its movement does not affect the result of *q*. Consequently, this method minimizes the communication cost between the moving objects and the server.

Although the methods proposed in Hu *et al*. (2005) and Mouratidis *et al*. (2005b) reduced the communication cost to a limited extent, they used many cellular links, which became the communication bottleneck in the system.

3 The proposed method

3.1 System overview

The system consists of a centralized server, a base station, and a large number of moving objects. Each moving object has two wireless interfaces, one for WiFi and the other for cellular networks (e.g., 3G cellular networks), which enable each moving object to operate in both the ad-hoc mode and the cellular mode.

The server can periodically broadcast information through a public wireless broadcasting channel accessible to all moving objects in the system. Also, the server plays a role of a mediator for monitoring tasks at the server side and the moving objects side. We assume synchronized time between the server and the moving objects. Consequently, the server can broadcast messages to the moving objects at the pre-scheduled time slots, while the moving objects periodically access the broadcasting channel at these time slots. Fig. 1 shows the general architecture of the system.

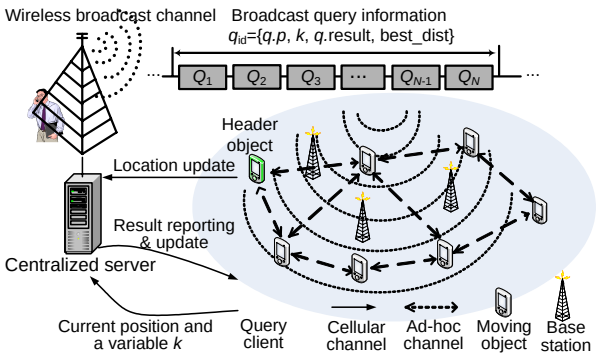

Fig. 1 General architecture of our proposed system for monitoring *k*NN queries

When a moving object *o* needs to report a location-update message to the server, instead of sending it directly to the server via the cellular uplink channel, *o* sends it to the designated moving object, referred to as the header object, via the WiFi link. The header object is responsible for (1) receiving the location-update messages from a number of moving objects, (2) aggregating these messages, and (3) sending them to the server via the cellular uplink channel. The routing of the messages from the moving objects to the header object is accomplished by using one of the location aided routing (LAR) protocols (Giordano *et al*., 2003).

3.2 Grid index and the notion of the influence region

The geographical area covered by the system is assumed to be a unit space, i.e., [0, 1)×[0, 1), and is mapped onto the regular grid of cells. The extent of a

cell on each dimension is $1/\delta$ so that the cell $c(i, j)$ at column i and row j (starting from the low-left corner of the grid) maintains the information of all the moving objects with x-coordinate in the range $[i\times(1/\delta), (i+1)\times(1/\delta)]$ and y-coordinate in the range $[j\times(1/\delta), (j+1)\times(1/\delta)]$. Note that δ is the system parameter set at the system initialization period. Among the moving objects in each cell $c(i, j)$, the object superior to the other objects in terms of computational capability and memory capacity is selected as the header object.

One important concept for monitoring continuous kNN queries is the 'influence region'. Intuitively, although there are numerous objects that move in arbitrary directions with arbitrary velocities in the system, it is sufficient to consider the objects whose movements may affect the results of some continuous kNN queries. For the remainder, we no longer need the up-to-date location information. According to the methodology of Babcock and Olston (2003) and Mouratidis *et al*. (2005b), we can avoid continually sending a large number of location-update messages. The influence region of each continuous kNN query q enables the possibility of minimizing the communication cost. It is the circular-shaped region centered at the query point $q.p$ with its radius being equal to best_dist. Here, best_dist is the distance from $q.p$ to the kth object in the current result of q. The influence region guarantees that the result of q is not changed if no location-update occurs inside this region. For example, Fig. 2a illustrates the initial influence region of a query q with its result being $\{o_1, o_2, o_3\}$. Assume that an object o_1 changes its location as depicted in Fig. 2b. Since the location-update occurs inside the influence region of q, the result of q is influenced, and thus it must be re-evaluated. After re-evaluation of q, a new influence region is assigned to o_1 (Fig. 2c).

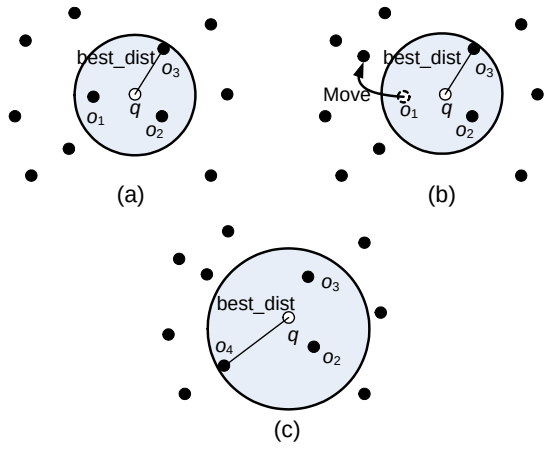

Fig. 2 Influence region of a query with three nearest neighbors

(a) Initial influence region; (b) Changed moving object; (c) Re-evaluated influenc region

3.3 The proposed *k*NN monitoring method

The whole monitoring procedure consists of two phases: the initial evaluation phase and the subsequent continual re-evaluation phase. In the initial evaluation phase, the server retrieves the initial result of a continuous kNN query. Then, the continual re-evaluation procedure, which keeps the query result up-to-date, is launched.

3.3.1 Initial evaluation

The server maintains the following data structures (Fig. 3):

Query table QT: QT stores queries with their coordinates, the values of k, current results, and the values of best_dist.

Grid index G: The server maps the geographical area onto grid G of cells, each of which is $\delta\times\delta$ square area. Each cell $c(i, j)$ of G is associated with (1) the list of continuous kNN queries whose influence regions intersect $c(i, j)$ and (2) the information of the header object that lies within $c(i, j)$.

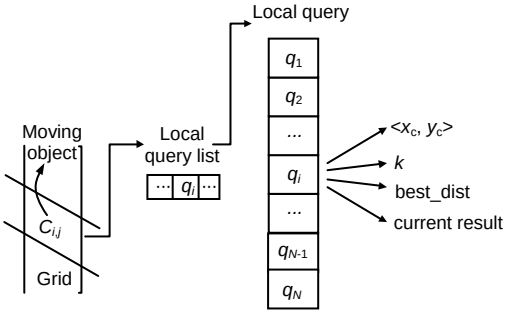

Fig. 3 Data structure maintained at the server

Once a continuous kNN query q is registered, the server first updates QT accordingly. Because the server does not know the initial result of q, it sets the result of q to $\varnothing$. Assuming the server knows only the cardinality $|O|$ of a set of moving objects $O=\{o_1, o_2, \cdots, o_{|O|}\}$ but not its distribution, it sets q.best_dist= $\sqrt{k/(\pi|O|)}$. The rationale is that for the uniform distribution of the moving objects in the unit space, the circular region centered at $q.p$ with its radius equal to

$\sqrt{k/(\pi|O|)}$ is expected to contain k objects (Bohm, 2000).

On each moving object o, a local query table o.lqt is maintained to store the information of the queries whose influence regions intersect the cell in which the object resides. Each entry in lqt has the following format: <q_i, q.coordinates, q.best_dist>, where q_i is the identifier of the query q. Based on q.coordinates and q.best_dist, o easily determines the influence region of q. Only the location-update affecting the influence region may change the result of q.

In existing methods, when a moving object o moves from the old cell to the new cell $c(i, j)$ of grid G, o contacts the server to request the queries whose influence regions currently intersect $c(i, j)$. In contrast, in the proposed method, the server periodically disseminates the query information through the dedicated broadcasting channel. Fig. 4 illustrates the broadcast stream. Instead of directly contacting the server, o just accesses the broadcast channel. Then, it downloads and monitors the associated queries, and sends the location-update message (if necessary) to the corresponding header object via the WiFi link. We use one of the LAR protocols (Ko and Vaidya, 1998) for simplicity and efficiency. The header object receives the location-update messages from a number of moving objects, aggregates these messages, and sends them to the server through the cellular uplink channel.

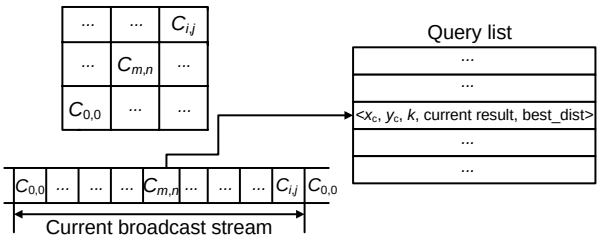

Fig. 4 Broadcast stream of query information

After receiving the aggregated messages from each header object in each cell, the server checks if, for each continuous kNN query q, the number of the objects that lie within the circular region, centered at $q.p$ with its radius equal to q.best_dist (e.g., $\sqrt{k/(\pi|O|)}$), is equal to or larger than k. If so, the server simply evaluates the result of q, and updates G since it has sufficient information. Otherwise, it sets larger q.best_dist based on the density information received from the header objects and updates the G accordingly. Note that the initial result can be evaluated when the server collects at least k moving objects. Algorithm 1 presents the algorithm of this process.

Algorithm 1 Initial evaluation phase

(a) Server side algorithm

Data structure: Query table QT

Input: Query q

1. Update QT;
2. Set q.result=∅;
3. Set q.best_dist= $\sqrt{k/(\pi|O|)}$;
4. Update broadcast stream S;

(b) Client side algorithm

Data structure: Local query table o.lqt

Input: A broadcast stream S

1. **if** it is not a header object **then**
2. **if** its movement affects some influence region **then**
3. Send a location-update message to the header object (via a WiFi channel);
4. **end if**
5. **else** // it is a header object
6. Aggregate messages from other moving objects;
7. Send the messages M to the server (via a cellular uplink channel);
8. **end if**

3.3.2 Continuous re-evaluation

The continuous re-evaluation phase for a query is launched after the initial result of the query is retrieved to keep the query result valid when the moving objects in the system cause a change to the current query result. The core idea of the continuous re-evaluation is to leverage the computational capability of the moving objects to avoid constant location-updates. The moving objects are allowed to monitor their movement directly against their associated queries. Each moving object performs the following process based on its movement:

If a moving object has moved within its previous cell $c(i, j)$, the moving object (1) accesses the broadcast channel and retrieves the information of the queries whose influence regions intersect $c(i, j)$, (2) checks the queries affected by its movement, and (3) notifies its new location (if necessary) to the header object in $c(i, j)$ using the WiFi link. The header object collects the location-update messages from moving objects and sends them to the server through the cellular uplink channel.

If a moving object has moved out of its previous cell $c(i, j)$, the moving object (1) accesses the broadcast channel and retrieves the information of the queries whose influence regions intersect the current cell $c(i', j')$, in which it resides, (2) checks the queries affected by its movement, and (3) transmits a location-update message to the header object in $c(i', j')$. Note that it is not necessary for the moving object to notify its movement to the header object in the previous cell $c(i, j)$.

After receiving the messages from the header objects, the server must update the result of each query q affected by the movement of the moving objects. Let $|o_{in}|$ and $|o_{out}|$ be the numbers of incoming objects (e.g., the outer objects that move into the influence region of q) and outgoing objects (e.g., the inner objects that move out of the influence region of q), respectively. For each query q in the query table QT, based on the cardinalities of $|o_{in}|$ and $|o_{out}|$, the server distinguishes the following two cases:

If $|o_{in}| \geq |o_{out}|$, the influence region of q guarantees to contain at least k objects. Therefore, the server can retrieve the new result incrementally without re-evaluation.

If $|o_{in}| < |o_{out}|$, the server re-evaluates q using the density information received from the header objects.

Finally, the server updates the grid G accordingly since the influence region of each query q may shrink or enlarge. Algorithm 2 presents the algorithm of this process.

Algorithm 2 Continual re-evaluation phases

(a) Server side algorithm

Data structure: Query table QT and grid index G
Input: Aggregated location-update messages M

```
1.  for each affected query q do
2.    if the number of moving objects in the previous
        influence region of q is greater than k then
3.      Retrieve the new result of q;
4.    else
5.      Retrieve the new result of q using the density
          information (easily obtained from M);
6.    end if
7.    Update G accordingly;
8.  end for
```

(b) Client side algorithm

Data structure: Local query table o.lqt
Input: A broadcast stream S

```
1.  if it has moved within its previous cell c(i, j) then
2.    Invoke (b) Client side algorithm in Algorithm 1;
3.  else // it has moved out of its previous cell c(i, j)
4.    if its movement affects some influence region then
5.      Send a location-update message to the header object
          in its current cell c(i', j') (via a WiFi channel)
6.    end if
7.  end if
```

4 Performance evaluation

In this section, we present the performance comparison of the proposed method, the threshold-based algorithm (Mouratidis *et al*., 2005b), and the centralized monitoring method (Mouratidis *et al*., 2005b; Xiong *et al*., 2005; Yu *et al*., 2005) with regard to the overall communication cost. The overall communication cost is defined as the sum of (1) the number of messages sent by the server to the moving objects and (2) the number of messages sent by the moving objects to the server. All the experiments were conducted on a Pentium IV 3.2 GHz machine with 1 GB RAM. The testbed for the performance comparison between the proposed method and the threshold based algorithm was implemented in Java language.

We conducted the performance analysis on a dataset consisting of moving objects and query points, generated by a pseudo-random number generator. The 64×64 grid index setting showed the best performance for all methods, and thus we set the granularity of the grid index G to 64×64. We varied (1) the number of moving objects from 10 000 to 100 000, (2) the number of queries from 1000 to 10 000, and (3) the value of k from 1 to 16 in the experiments. Table 1 illustrates the parameter settings used in the experiments.

Table 1 Parameter settings used in the experiments

Parameter	Setting
Number of moving objects	10k, 20k, 40k, 80k, 100k
Number of queries	1k, 2k, 4k, 8k, 10k
Value of k	1, 2, 4, 8, 16

Fig. 5 illustrates the impact of the number of moving objects on the communication cost. We set the number of queries to 4k and the value of k to 4, and measured the communication cost by varying the number of moving objects. In the centralized method, the communication cost was proportional to the

number of moving objects since each moving object periodically sends a location-update message to the server. In contrast, the communication costs of both methods slightly increased. The proposed method outperformed the threshold method. On average, the proposed method incurred 88% of the communication cost of the threshold method.

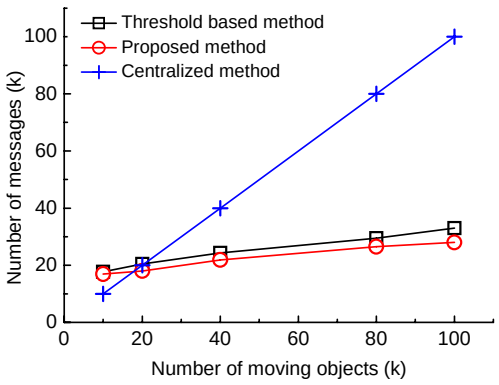

Fig. 5 Effect of the number of moving objects on the communication cost with 4000 queries and k=4

Next, we set the number of moving objects to 40k and the value of k to 4, and measured the communication cost by varying the number of queries (Fig. 6). Note that the number of queries did not affect the communication cost of the centralized method since each moving object periodically reports its location without considering the query information. In contrast, the communication costs of the proposed method and the threshold based algorithm increased as the number of queries increased. As expected, the proposed method achieved a better performance than the threshold based method. This is due to the fact that the proposed method uses the WiFi link as well as the wireless broadcasting strategy. The proposed method incurred 75% of the communication cost of the threshold method.

Finally, we studied the effect of k on the performance of the methods (Fig. 7). We set the number of queries to 4k and the number of moving objects to 40k, and measured the communication cost by varying the value of k. Again, the value of k did not affect the communication cost of the centralized method for the same reason as above stated. As k increased, the performances of both the proposed method and threshold based algorithm deteriorated. This is because the larger k leads to the more query results being changed. The proposed method clearly outperformed the threshold based algorithm. In comparison with the threshold based algorithm, the proposed method required only 63% of the communication cost on average.

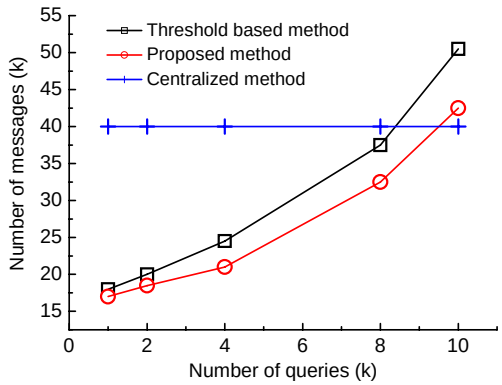

Fig. 6 Effect of the number of queries on the communication cost with 40 000 moving objects and k=4

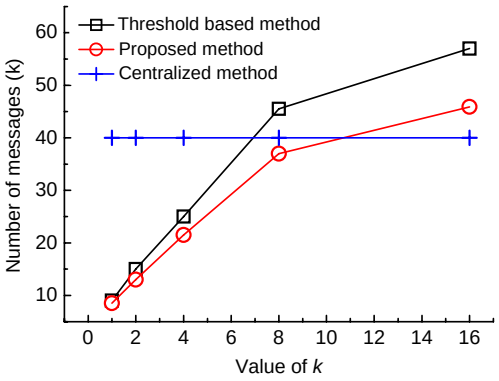

Fig. 7 Effect of the value of k on the communication cost with 4000 queries and 40 000 moving objects

5 Conclusions

In this paper, we address the problem of monitoring continuous kNN queries on moving objects in a distributed approach. Given a large number of clients, the main purpose of our work is to continuously manage the result of each client with minimum communication cost. Since wireless broadcasting has been widely accepted for disseminating information to a large number of moving objects, we assume an environment where moving objects construct ad hoc networks in the wireless data broadcasting system, and propose the method that leverages the WiFi link

to reduce the burden on the cellular uplink communication cost. Simulation results showed the efficiency of the proposed method. We expect that monitoring continuous *k*NN queries will play a critical role in a variety of real-life applications in mobile/pervasive computing environments. For future work, we plan to extend our proposed method to other query types like reverse nearest neighbor queries.

References

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E., 2002. A survey on sensor network. *IEEE Commun. Mag.*, **40**(8):102-114. [doi:10.1109/MCOM.2002.1024422]

Babcock, B., Olston, C., 2003. Distributed Top-*k* Monitoring. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.28-39. [doi:10.1145/872757.872764]

Bohm, C., 2000. A cost model for query processing in high dimensional data spaces. *ACM Trans. Database Syst.*, **25**(2):129-178. [doi:10.1145/357775.357776]

Cheung, K.L., Fu, A.W.C., 1998. Enhanced nearest neighbor search on the R-tree. *ACM SIGMOD Rec.*, **27**(3):16-21. [doi:10.1145/290593.290596]

Giordano, S., Stojmenovic, I., Blazevic, L., 2003. Position-Based Routing Algorithms for Ad Hoc Networks: a Taxonomy. *In*: Cheng, X., Huang, X., Du, D.Z. (Eds.), Ad Hoc Wireless Networking. Kluwer Academic Publishers, Dordrecht, p.103-136.

Guttman, A., 1984. R-trees: a dynamic index structure for spatial searching. *ACM SIGMOD Rec.*, **14**(2):47-57. [doi:10.1145/971697.602266]

Hjaltason, G.R., Samet, H., 1999. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, **24**(2):265-318. [doi:10.1145/320248.320255]

Hu, H., Xu, J., Lee, D., 2005. A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.479-490. [doi:10.1145/1066157.1066212]

Imielinski, T., Viswanathan, S., Bardrinath, B.R., 1997. Data on air: organization and access. *IEEE Trans. Knowl. Data Eng.*, **9**(3):353-372. [doi:10.1109/69.599926]

Junglas, I.A., Watson, R.T., 2008. Location-based services. *Commun. ACM*, **51**(3):65-69. [doi:10.1145/1325555.1325568]

Ko, Y.B., Vaidya, N.H., 1998. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. Proc. 4th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking, p.66-75. [doi:10.1145/288235.288252]

Lee, D.L., Lee, W.C., Xu, J., Zheng, B., 2002. Data management in location-dependent information services: challenges and issues. *IEEE Perv. Comput.*, **1**(3):65-72. [doi:10.1109/MPRV.2002.1037724]

Mouratidis, K., Hadjieleftheriou, M., Papadias, D., 2005a. Conceptual Partitioning: an Efficient Method for Continuous Nearest Neighbor Monitoring. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.634-645. [doi:10.1145/1066157.1066230]

Mouratidis, K., Papadias, D., Bakiras, S., Tao, Y., 2005b. A threshold-based algorithm for continuous monitoring of *k* nearest neighbors. *IEEE Trans. Knowl. Data Eng.*, **17**(11):1451-1464. [doi:10.1109/TKDE.2005.172]

Papadopoulos, A., Manolopoulos, Y., 1997. Performance of Nearest Neighbor Queries in R-Trees. Proc. 6th Int. Conf. on Database Theory, p.394-408.

Roussopoulos, N., Kelley, S., Vincent, F., 1995. Nearest neighbor search. *ACM SIGMOD Rec.*, **24**(2):71-79. [doi:10.1145/568271.223794]

Tao, Y., Papadias, D., 2002. Time-Parameterized Queries in Spatio-Temporal Databases. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.334-345. [doi:10.1145/564691.564730]

Xiong, X., Mokbel, M.F., Aref, W.G., 2005. SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio-Temporal Databases. 21st Int. Conf. on Data Engineering, p.643-654. [doi:10.1109/ICDE.2005.128]

Yu, X., Pu, K.Q., Koudas, N., 2005. Monitoring k-Nearest Neighbor Queries over Moving Objects. 21st Int. Conf. on Data Engineering, p.631-642. [doi:10.1109/ICDE.2005.92]

Zhang, J., Zhu, M., Papadias, D., Tao, Y., Lee, D., 2003. Location-Based Spatial Queries. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.443-454. [doi:10.1145/872757.872812]