ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Unremitting Monitoring of Spatial Queries in Wireless Network Environments

D. Raghu
Professor in Computer Science Department
Nova College of Engineering & Technology
Jangareddigudem, West Godavari District, A.P., India
raghuau@gmail.com

MD. Sajid Pasha*
Dept. of Computer Science
Nova College of Engineering & Technology,
Jangareddigudem, , West Godavari District, A.P., India,
sajnajsam@gmail.com

Ch. Raja Jacob
Associate Professor, Dept. of Computer Science,
Nova College of Engineering & Technology,
Jangareddigudem, West Godavari District, A.P., India
rchidipi@gmail.com

Abstract: Wireless data broadcast is a promising technique for information dissemination that leverages the computational capabilities of the mobile devices in order to enhance the scalability of the system. Under this environment, the data are continuously broadcast by the server, interleaved with some indexing information for query processing. Clients may then tune in the broadcast channel and process their queries locally without contacting the server. Previous work on spatial query processing for wireless broadcast systems has only considered snapshot queries over static data. In this paper, we propose an air indexing framework that

1) Outperforms the existing (i.e., snapshot) techniques in terms of energy consumption while achieving low access latency and
2) Constitutes the first method supporting efficient processing of continuous spatial queries over moving objects.

Key words: Spatial database, query processing, location based service, wireless data broadcast, air index

I. INTRODUCTION

Mobile devices with computational, storage, and wireless communication capabilities (such as PDAs) are becoming increasingly popular. At the same time, the technology behind positioning systems is constantly evolving, enabling the integration of low-cost GPS devices in any portable unit. Consequently, new mobile computing applications are expected to emerge, allowing users to issue location-dependent queries in a ubiquitous manner. Consider, for instance, a user (mobile client) in an unfamiliar city, who would like to know the 10 closest restaurants. This is an instance of a k nearest neighbor (kNN) query, where the query point is the current location of the client and the set of data objects contains the city restaurants. Alternatively, the user may ask for all restaurants located within a certain distance, i.e., within 200 meters. This is an instance of a range query. Spatial queries have been studied extensively in the past and numerous algorithms exist for processing snapshot queries on static data indexed by a spatial access method. Subsequent methods [22], [24], [30] focused on moving queries (clients) and/or objects. The main idea is to return some additional information eg. [10],[11],[21] more NNs, expiry time, validity region that determines the lifespan of the result.

Thus, a moving client needs to issue another query only after the current result expires. These methods focus on single query processing, make certain assumptions about object movement (e.g., static in [22], [30], linear in [24]), and do not include mechanisms for maintenance of the query results (i.e., when the result expires, a new query must be issued).

Recent research considers continuous monitoring of multiple queries over arbitrarily moving objects. In this setting, there is a central server that monitors the locations of both objects and queries. The task of the server is to report and continuously update the query results as the clients and the objects move.

In the aforementioned methods, the processing load at the server side increases with the number of queries. In applications involving numerous clients, the server may be overwhelmed by their queries or take prohibitively long time to answer them. To avoid this problem, Imielinski et al. [14] propose wireless data broadcast, a promising technique that leverages the computational capabilities of the clients' mobile devices and pushes the query processing task entirely to the client side. In this environment, the server only monitors the locations of the data objects, but is unaware of the clients and their queries. The data objects are continuously broadcast by the server, interleaved with some indexing information. The clients utilize the broadcast index, called air index, to tune in the channel only during the transmission of the relevant data and process their queries locally. Thus, the server load is independent of the number of clients. Previous work on location-dependent spatial query processing for wireless broadcast systems has only considered snapshot queries over static data.

II. RELATED WORK

The transmission schedule in a wireless broadcast system consists of a series of broadcast cycles. Within each cycle the data are organized into a number of index and data buckets. A bucket (which has a constant size) corresponds to the smallest logical unit of information, similar to the page concept in conventional storage systems. A single bucket may be carried into multiple network packets (i.e., the basic unit of information that is transmitted over the air). However, they are typically assumed to be of the same size (i.e., one bucket equals one packet). The most common data organization method is the (1,m) interleaving scheme [14],

as shown in the below figure. The data objects are divided into m distinct segments and each data segment in the transmission schedule is preceded by a complete version of the index. In this way, the access latency for a client is minimized, since it may access the index.

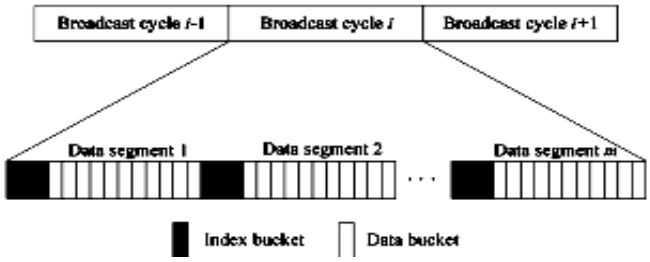

Figure 1

A. The (1, m) interleaving scheme:

The main motivation behind air indexes is to minimize the power consumption at the mobile client. Although in a broadcast environment, the uplink transmissions are avoided, receiving all the downlink packets from the server is not energy efficient. For instance, the Cabletron 802.11 network card (wireless LAN) was found to consume 1,400 mW in the transmit, 1,000 mW in the receive, and 130 mW in the sleep mode [5]. Therefore, it is imperative that the client switches to the sleep mode (i.e., turns off the receiver) whenever the transmitted packets do not contain any useful information. Based on the data organization technique in Fig. 2, the query processing at the mobile client is performed as follows: 1) the client tunes in the broadcast channel when the query is issued and goes to sleep until the next index segment arrives, 2) the client traverses the index and determines when the data objects satisfying its query will be broadcast, and 3) the client goes to sleep and returns to the receive mode only to retrieve the corresponding data objects.

Most relevant to our work are the techniques related to kNN search on the air. Zheng et al. [31] propose an approximate kNN query processing algorithm that is not guaranteed to always return k objects. The idea is to use an estimate r of the radius that is expected to contain at least k points. Using this estimate, the search space can be pruned efficiently at the beginning of the search process.

III. SNAPSHOT KNN QUERIES

In snapshot kNN queries we use the following indexes;

i. Air index structure
ii. Query processing

A. Air index structure:

BGI indexes the data objects with a regular grid, i.e., a partitioning of the data space into square cells of equal size with side-length _ (a system parameter). Each cell stores the object coordinates falling inside and maintains their total number. Consider Fig. 5a, where the data objects in the system are p1 to p20, and is set as shown. In the example, cell c0;0 contains the coordinates of objects p1 and p2 and δ has cardinality 2. Given an object with coordinates x and y, its covering cell is ci;j, where I=[x/δ] and j=[y/δ]. Similarly, given a cell ci;j, its corresponding region is [i. δ, i. δ+δ] x [j. δ,j.+ δ]. The grid information is placed into packets to form the index segment of BGI. Note that the index segment contains only1 the object coordinates to keep its size small. Following the ð1;mÞ scheme, the full object information is broken into m data segments, each preceded by a copy of the index segment. The value of m is determined using the analysis of [14].

B. Query Processing:

The kNN computation runs completely at the client side. Let q be the client location. Given a cell c; maxdist(c) is the maximum possible distance between any point in c and q. If the cardinality of c is c.card, then at least c.card objects lie within distance maxdist(c) from q. Similarly, mindist(c) is the minimum possible distance between any point in c and q . If there are at least k objects within distance dmax from q, then a cell c (or bucket) does not have to be considered if mindist(c) dmax, since it cannot contain any better neighbor. Based on the above observations, the NN computation algorithm consists of two steps. During the first step, the client receives (some) upper level buckets. According to the cardinalities and the maxdist of the contained cells, it computes a conservative upper bound dmax of the radius around q that contains at least k objects. During the second step, the client listens to the contents of cells c (in the lower level) that have mindist(c) < dmax; cells (and buckets) with mindist above dmax are skipped. After the second step, the client already knows the coordinates and the packets containing the full information of its kNNs. An important remark is that during each step, the bound d_{max} keeps decreasing, excluding more unnecessary packets from consideration.

IV. ALOGRITHMS USED

Consider, for instance, a user (mobile client) in an unfamiliar city, who would like to know the 10 closest restaurants. This is an instance of a k nearest neighbor (kNN) query, where the query point is the current location of the client and the set of data objects contains the city restaurants. Alternatively, the user may ask for all restaurants located within a certain distance, i.e., within 200 meters. This is an instance of a range query.

Most relevant to our work are the techniques related to kNN search on the air. Zheng propose an approximate kNN query-processing algorithm that is not guaranteed to always return k objects. The idea is to use an estimate r of the radius that is expected to contain at least k points. Using this estimate, the search space can be pruned efficiently at the beginning of the search process.

The authors also introduce a learning algorithm that adaptively reconfigures the estimation algorithm to reflect the distribution of the data. Regarding the query processing phase, two different approaches are proposed:

A. The standard R- tree index enhanced with the aforementioned pruning criterion and

A new-sorted list index that maintains a sorted list of the objects on each spatial dimension.

The sorted list method is shown to be superior to the R-tree only for small values of k. Gedik describe several algorithms to improve kNN query processing in sequential-access R-trees. They investigate the effect of different broadcast organizations on the tuning time and also propose the use of histograms to enhance the pruning capabilities of the search algorithms. Park focus on reducing the access latency of kNN search by accessing the data segment of the broadcast cycle. In particular, they propose a method where the data objects are sorted according to one spatial coordinate. In this way, the client does not need to wait for

the next index segment to arrive, but can start query processing immediately by retrieving the actual data objects.

B. Knn alogrithm on static data:

a. kNN computation:

// Client at q goes online, and listens to the first index segment
// Step 1: The upper level is broadcast
i. best_NN = ∅; $d_{max} = \infty$
ii. **for** each bucket B
iii. **if** minidist (B) < d_{max} // Prune upper level buckets
iv. **for** each cell c in B
v. **for** iter = 1 to c.card
vi. **if** maxidst (c) < d_{max}
vii. Delete the kth entry of best_NN
viii. Insert <c, maxdist(c) > to best_NN; Update d_{max}
// Step 2: The lower level is broadcast
ix. **for** each cell c with mindist(c) < d_{max}
x. Delete all entries of c from best_NN
xi. **for** each object p in c
xii. **if** dist(p) ≤ d_{max}
xiii. Delete the kth entry of best_NN
xiv. Insert < p, dist(p) > into best_NN; Update d_{max}
xv. **return** best_NN

C. Knn algorithm on moving data:

// Client at q has completed the kNN computation
// and maintains a list L of dirty grid cell cardinalities
i. Listen to the contents of the next dirty grid
ii. **for** each dirty grid cell c with its bit set
iii. Delete c from L
iv. **if** no dirty cell overlaps with circ(q) and $q^1 = q$
v. **return** // result set has not changed
vi. else
vii. best_NN = ∅ ; $d_{max} = \infty$
viii. **for** each object p in the current result
ix. **if** p's cell is not dirty
x. insert <p, dist(p,q^1) > to best_NN
xi. **for** each cell c in L
xii. **for** iter = 1 to c.card
xiii. **if** maxdist(c) < d_{max}
xiv. Delete the kth entry of best_NN
xv. Insert <c, maxdist(c) > to best_NN; Update d_{max}
xvi. Invoke the kNN computation algorithm
xvii. using the current value of d_{max}
xviii. **return**

V. EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed methods under various system parameters. We use a real 4 spatial data set (REAL) containing the locations of 5,848 cities and villages in Greece (available at www.rtreeportal. org). Additionally, in order to investigate scalability, we generated five skewed data sets (SKEW), where the object locations follow a Zipf distribution with parameter 0.8.

Table I

Parameter	Range
SKEW DB size	10K, 50K **100K,** 150K, 200K
Packet size (bytes)	64, 128, **256,** 512, 1024
Range query area (km^2)	12.5, 25, **50,** 100, 200, 400
Number of NNs (k)	1, 2, **4,** 8, 16, 32
Object / query speed (km/h)	5, **25,** 125
Object / query agility (%)	0, 10, 20, 30, 40, **50,** 60, 70, 80, 90,100

All data sets are scaled to fit in a [0, 10,000]2 workspace. Assuming that the data objects are distributed on a 50 km _ 50 km area, the average density of the objects varies between 2.3 and 80 objects/km2. For static data, each object corresponds to a point in the data set. For generating moving data, we randomly select the initial position and the destination of each object from the spatial data set. The object then follows a linear trajectory (with constant velocity) between the two points. Upon reaching the endpoint, a new random destination is selected and the same process is repeated. Distance is defined according to the euclidean metric. To further control the object movement, only a certain fraction (which we call agility) of the objects moves during each time stamp. The same pattern is also adopted for the moving queries.

VI. CONCLUSION

In this paper, we study spatial query processing in wireless broadcast environments. A central server transmits the data along with some indexing information. The clients process their queries locally, by accessing the broadcast channel. In this setting, our target is to reduce the power consumption and access latency at the client side. We propose an on-air indexing method that uses a regular grid to store and transmit the data objects. We design algorithms for snapshot and continuous queries, over static or dynamic data. To the best of our knowledge, this is the first study on air indexing that 1) addresses continuous queries and 2) considers moving data objects. We demonstrate the efficiency of our algorithm through an extensive experimental comparison with the current state-of-the-art frameworks for snapshot queries and with the naïve constant recomputation technique for continuous queries. A challenging problem is to devise cost models for continuous monitoring of spatial queries in wireless broadcast environments. Such models could reveal the best technique given the problem settings, help fine-tune several system parameters (e.g., grid size), and potentially lead to better algorithms. Another interesting direction for future work is to study different types of spatial queries, such as reverse nearest neighbors, and to extend our framework to process their snapshot and continuous versions.

VII. REFERENCES

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

[2] S. Acharya, M.J. Franklin, and S.B. Zdonik, "Disseminating Updates on Broadcast Disks," Proc. Int'l Conf. Very Large Data Bases (VLDB '96), 1996.

[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, 1990.

[4] Y. Cai, K.A. Hua, and G. Cao, "Processing Range-Monitoring Queries on Heterogeneous Mobile Objects," Proc. IEEE Int'l Conf. Mobile Data Management (MDM '04), 2004.

[5] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," Proc. ACM MobiCom, 2001.

[6] M.-S. Chen, P.S. Yu, and K.-L. Wu, "Indexed Sequential Data Broadcasting in Wireless Mobile Computing," Proc. Int'l Conf. Distributed Computing Systems (ICDCS '97), 1997.

[7] B. Gedik and L. Liu, "MobiEyes: Distributed Processing of Continuously Moving Queries on Moving Objects in a Mobile System," Proc. Int'l Conf. Extending Database Technology (EDBT '04), 2004.

[8] B. Gedik, A. Singh, and L. Liu, "Energy Efficient Exact kNN Search in Wireless Broadcast Environments," Proc. Ann. ACM Int'l Workshop Geographic Information Systems (GIS '04), 2004.

[9] S.E. Hambrusch, C.-M. Liu, W.G. Aref, and S. Prabhakar, "Query Processing in Broadcasted Spatial Index Trees," Proc. Int'l Symp. Advances in Spatial and Temporal Databases (SSTD '01), 2001.

[10] Henrich, "A Distance Scan Algorithm for Spatial Access Structures," Proc. Second ACM Workshop Geographic Information Systems (GIS '94), 1994.

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

[12] Q. Hu, W.-C. Lee, and D.L. Lee, "Power Conservative Multi- Attribute Queries on Data Broadcast," Proc. Int'l Conf. Data Eng.(ICDE '00), 2000.

[13] T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Power Efficient Filtering of Data an Air," Proc. Int'l Conf. Extending Database Technology (EDBT '94), 1994.

[14] 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 1997.

[15] D.V. Kalashnikov, S. Prabhakar, and S.E. Hambrusch, "Main Memory Evaluation of Monitoring Queries over Moving Objects," Distributed and Parallel Databases, vol. 15, no. 2, pp. 117-135, 2004.

[16] Kamel and C. Faloutsos, "On Packing R-Trees," Proc. Conf. Information and Knowledge Management (CIKM '93), 1993.

[17] W.-C. Lee and B. Zheng, "DSI: A Fully Distributed Spatial Index for Location-Based Wireless Broadcast Services," Proc. Int'l Conf. Distributed Computing Systems (ICDCS '05), 2005.

[18] M.F. Mokbel, X. Xiong, and W.G. Aref, "SINA: Scalable Incremental Processing of Continuous Queries in Spatio-Temporal Databases," Proc. ACM SIGMOD, 2004.

[19] K. Mouratidis, M. Hadjieleftheriou, and D. Papadias, "Conceptual Partitioning: An Efficient Method for Continuous Nearest Neighbor Monitoring," Proc. ACM SIGMOD, 2005.

[20] K. Park, M. Song, and C.-S. Hwang, "An Efficient Data Dissemination Schemes for Location Dependent Information Services," Proc. Int'l Conf. Distributed Computing and Internet Technologies (ICDCIT '04), 2004.

[21] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," Proc. ACM SIGMOD, 1995.

[22] Z. Song and N. Roussopoulos, "K-Nearest Neighbor Search for Moving Query Point," Proc. Int'l Symp. Spatial and Temporal Databases (SSTD '01), 2001.

[23] Stojmenovic, Handbook of Wireless Networks and Mobile Computing. John Wiley & Sons, 2002.

[24] Y. Tao and D. Papadias, "Spatial Queries in Dynamic Environments," ACM Trans. Database Systems, vol. 28, no. 2, pp. 101-139, 2003.

[25] Y. Tao, D. Papadias, and Q. Shen, "Continuous Nearest Neighbor Search," Proc. Conf. Very Large Data Base (VLDB '02), 2002.

[26] Xiong, M.F. Mokbel, and W.G. Aref, "SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio- Temporal Databases," Proc. Int'l Conf. Data Eng. (ICDE '05), 2005.

[27] J. Xu, W.-C. Lee, and X. Tang, "Exponential Index: A Parameterized Distributed Indexing Scheme for Data on Air," Proc. MobiSys, 2004.

[28] J. Xu, B. Zheng, W.-C. Lee, and D.L. Lee, "Energy Efficient Index for Querying Location-Dependent Data in Mobile Broadcast Environments," Proc. Int'l Conf. Data Eng. (ICDE '03), 2003.

[29] X. Yu, K.Q. Pu, and N. Koudas, "Monitoring K-Nearest Neighbor Queries over Moving Objects," Proc. Int'l Conf. Data Eng. (ICDE '05), 2005.

[30] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D.L. Lee, "Location- Based Spatial Queries," Proc. ACM SIGMOD, 2003.