Study on RNN Query in Broadcasting Environment

Lien-Fa Lin*, Chao-Chun Chen
Department of Computer Science and Information Engineering National Cheng-Kung University, Tainan, Taiwan, R.O.C.
Department of Information Communication Southern Taiwan University of Technology, Tainan, Taiwan, R.O.C.

lienfa@cc.kyu.edu.tw,chencc@mail.stut.edu.tw

ABSTRACT

Location-based services (LBSs) provide information based on location information specified in a query. Queries that support for LBS are called Location-Dependent Queries (LDQ). One such query is the Reverse Nearest Neighbor (RNN) query that returns the objects that have a query object as their closest object. Just like the Nearest Neighbor (NN) queries, the RNN queries appear in many practical applications such as decision support system, continuous referral systems, profile-based marketing, maintaining document repositories, bioinformatics, etc. Thus efficient methods for the RNN queries in database are required. While the RNN is well studied in the traditional wired, disk-based client-server environment, it has not been tackled in a wireless broadcasting environment. The liner property of wireless broadcast media and power conserving requirement of mobile devices make the problem particularly interesting and challenging. In this paper, the issues involved with organizing location dependent data and answering RNN queries on air are investigated. An efficient data organization, called Jump Rdnn-tree, and the corresponding search algorithms are proposed. Performance of the proposed Jump Rdnn-tree and other traditional indexes (enhanced for wireless broadcast) is evaluated using both uniform and skew data. The results show that Jump Rdnn-tree substantially outperforms the traditional indexes.

Keywords: location-dependent services, data broadcast, energy-conserving, mobile computing

1 INTRODUCTION

Owing to the popularity of personal digital devices and advances in wireless communication technologies, location-based services (LBSs) have received a lot of attention from both of the industrial and academic communities [6,11,12,16,17,18]. With the maturation of necessary technologies and the anticipated world wide deployment of 3G wireless communication infrastructure, LBSs are essential applications in wireless networking environment. The query concerns LBSs, we called it LDQ (location-dependent query). The LDQ's applications contains range query, nearest neighbor (NN), k-nearest neighbor (KNN) query and reverse nearest neighbor (RNN) query etc.

In the past study on LDQ includes NN [5,15] query, KNN [3,4,8,12] query, CNN [21,23] query and CKNN [22,23] query are abundant and successful. And in recent years, the researchers have considerable attention on RNN query questions too. The query concerns LBSs, we called it LDQ (location-dependent query).[1]

The RNN problem has been introduced in database setting by Korn and Muthukrishman [9] along with several applications. For example, the bank plans to establish a new branch. If customers always prefer the nearest branch, then the new branch should be established on the location where the distance to such location for the majority of customers is shorter than that to other banks. Another common example is how a taxi driver chooses customers. By using wireless devices, a taxi driver may know the location of a customer who is looking for a taxi. From the view of competition, RNN is

* Lien-Fa Lin is also a lecturer at the Department of Information Communication, Kao Yuan University, No.1821, JhongShan Rd, Lujhu Township, Kaohsiung Country 821, Taiwan, R.O.C.

more meaningful than NN. As shown in Figure 1, the nearest neighbor to Taxi A is Customer C, but that does not necessarily mean Taxi A is the most likely to get to Customer C because Taxi B is even closer to Customer C. On the contrary, Taxi A should head for Customer D because Taxi A is the nearest neighbor in relation to Customer D. That is, the RNN for Taxi A is Customer D, and Taxi A may get to Customer D faster than all other taxis.

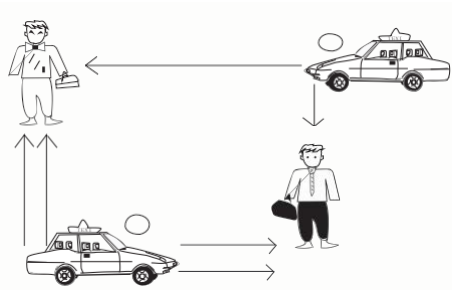


Figure 1: Example of RNN query application

As mobile device users increase, it has become a great challenge to availability of LBSs regardless of the increasing number of users. Wireless broadcasting technology is a solution to this problem [1,4,7]. Data delivery via broadcasting channels allows any number of mobile users (MU) to receive data at the same time. In addition, to effectively conserve power of mobile devices, the common practice is to broadcast data and on air index through broadcasting channels in an interlaced fashion. With on air index, MU knows when the required data will be broadcasted; the doze mode of mobile device, therefore, can be selected first, and then the active mode can be switched on until the arrival of the required data without wasting power by maintaining active mode to wait for the arrival of required data. The studies of using on air index technology, making MU use selective tuning to conserve power are plenty and popular [3,4,8,11,12,15,20].

Effective power conservation for mobile devices in wireless environment is a critical issue. Therefore, there is much literature dedicated to general query processing on mobile devices with effective power management [13,14,15,18,20]. From these studies we have deduced some principles for designing a good on air index. We use these principles to design an on air index method that can process RNN query efficiently. In addition, simulation experiments proved that our method may significantly improve efficiency when compared to Rdnn-tree modified for broadcasting environment.

The rest of paper is organized as follows. Section 2 is an overview of related work. In Section 3, we describe the effectiveness on air indexing design rules. The details of Jump Rdnn-tree index structure are introduced in Section 4. In Section 5, we describe the experiment environment. Performance results are shown in Section 6. Finally, we summarize the paper and describe our future work in Section 7.

2 RELATED WORK

In this section, we shall introduce RNN query, and research topics that relate to on air index and RNN query in broadcasting environment in the following subsections

2.1 Reverse Nearest Neighbor Query

The so-called RNN query that means offers a certain objects set S and a query object *q* to find out the objects which *q* is their nearest neighbor (NN) object. The RNN query application is quite widespread, including decision-making support system, biological information and so on. In [5,9,10] mention many about the RNN query application example.

A straightforward solution to computing reverse nearest neighbor (RNN) queries is to check for each point whether it has a given query point as its nearest neighbor. However, this method is not practical when processing large amount of data, because the time complexity involved is O (N^3). Therefore, the general method is use a specially R-tree (is called RNN tree [9]) to process query. Conjun Yang [5] proposed Rdnn-tree index structure to improve the method in [9]. This Rdnn-tree index structure can be applied to solve NN and RNN query problem simultaneously. The difference between Rdnn-tree and R-tree is that Rdnn-tree has recorded each object's NN information which can be used to process RNN query effectively.

2.2 Wireless Data Broadcas

MU may access LBSs information in wireless broadcasting environment with two methods:

On-demand Access: MU submits query to server, and server may use disk-based spatial index to accelerate query processing and increase data access efficiency. Server side is responsible to filter out data requested by MU and return the result to MU.

Broadcast and Filter: Data is broadcasted on public wireless channels periodically. MU simply tunes into the broadcast channel to access required data instead of constantly submitting query to server.

On-demand access uses basic client-server model, where server is responsible for query processing and returning results to users via point-to-point dedicate channel. However, on-demand access is more adequate on the system with less contention for wireless bandwidth, server processing, and workload. When the number of users increases,

system efficiency will reduce rapidly. As for wireless broadcasting applied to the radio and TV industry, the workload of a server is same and isn't affected by the number of users; the server still delivers one set of data only. It is a natural solution for user scalability and bandwidth problems.

On top of that, because mobile devices have a very limited supply of power, efficient power conservation is a major issue for mobile devices in wireless environment. In order to conserve power, it is common that mobile device design includes operation modes of active mode and doze mode [20]. A typical wireless PC card consumes 60mW in doze mode and 805–1400mW in active mode [16]. Power conservation in wireless broadcasting environment is achieved by adding index data to broadcasted data. By querying index data, users may know the time when required data will be broadcasted and select doze mode to save power and turn to active mode to access required data when schedule broadcast time is on.

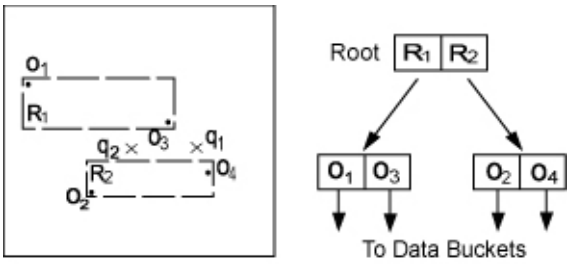
2.3 On Air Index

In traditional disk-based access environment, back-tracking is often used in query algorithm to enhance query efficiency. However, it makes problems if used in broadcast channels where only linear access is available.

In a wireless broadcasting environment, users may access data only when index data is being broadcasted. Therefore, when the sequence that the algorithm obtains index data is opposite to that of broadcasting, users must wait until the next broadcasting of such index data. For traditional database, on the contrary, index data that is stored on resident storage media, such as disk or memory chip, can be accessed in any time.

Because linear access is not considered in the design of traditional index structure, the algorithm that is currently adopted in disk-based spatial index can not satisfy the need of effective power conservation. Shown in Figure 2 is R-tree index; its broadcasting sequence is root, R_1, and R_2. The visit sequence for searching for NN with a given query point of q_2 is shown in Figure 3 (a). Root is first visited because the distance between q_2 and R_2 is shorter than that to R_1. Therefore, R_1 is skipped and R_2 is visited first. However, the shortest object to q_2 is o_3 of R_1 in MBR, and therefore R_1 must be first visited. However, at this time R_1 has just been broadcasted and it can only be accessed in the next broadcasting cycle. With the feature of linear access in broadcasting environment, if the broadcasting sequence differs from the sequence of query, then long access latency will occur. Therefore, branch-and-bound query method in broadcasting environment is not a very effective method in term of access latency. An alternative, as shown in Figure 3 (b), is direct access to MBRs sequentially. However, this method will cause unnecessary traversal of MBRs, and index search performance will not be optimized. For example, the search of NN for q_1, the real NN is O_4 of MBR R_2, and accessing to R_1 is obviously a waste of resource. Therefore, a new index method must be designed for wireless broadcasting environment to effectively adopt the feature of linear access in broadcasting environment and satisfy the need of power conservation for mobile devices.

In this paper, we have proposed a set of better broadcast index design principles and a modified Rdnn-Tree structure by adding the so-called jump pointer to make index tree accommodate linear access and to eliminate several unnecessary indexes to shrink the size of index data. We call this new index structure Jump-Rdnn Tree.



(a) MBR structure. (b) R-tree index

Figure 2: A running example of R-tree Indexing.

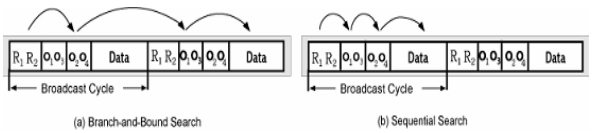


Figure 3: Linear access in wireless broadcasting environment

3 EFFECTIVE BROADCAST INDEX DESIGN

Access latency of accessing to data and tuning time that a mobile device requires in active mode are the two benchmarks for broadcast index efficiency measurements. Access latency is the time required for accessing to data from the moment a user gives the query command to the data that satisfies the query is accessed. Tuning time is the time required for users to receive requested data in active mode. Broadcast index is mixed with broadcast data and sent out together, and MU receives data in the following three steps [12]:

(1) Initial probe: during any point in time of broadcasting, a user tunes into a broadcast channel and wait for the index data to be broadcasted. This

period of time is called initial probe waiting.

(2) Index search: When index data arrives, a user receives the index data, selectively accesses some index data according to his/her needs, and finds the location of the requested data.

(3) Data retrieval: When the requested data arrives, a user downloads and accesses to the data. The time required for these three steps shall influence broadcast index efficiency. Therefore, a design of effective broadcast index must reduce the time required for these three steps.

Reduction of initial probe time: Initial probe waiting is the time that a user waits for index data. By duplicating multiple indexes in the entire broadcast cycle, the possibility of index appearing may increase, and the initial probe waiting time can be reduced. Imielinski et al. [7] used interleaving method, as shown in Figure 4 (1: m), to duplicate m copies of index data in order to reduce initial probe waiting time.

Reduction of index data size: Index searching time is related to the size of index data; the smaller the index data size is, the shorter the search time will be. Consequently, the entire broadcast cycle will be shorter, and the average access latency will be smaller. For example, Imielinski et al. [20] only duplicated k layers of index tree to reduce the size of index data. Hu et al. [18] used the signature capture technique to reduce index data size.

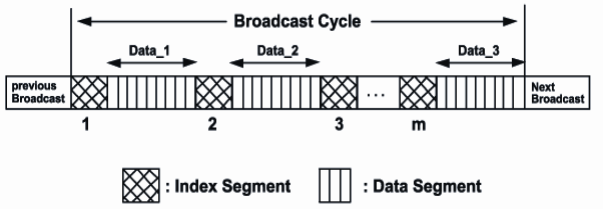


Figure 4: Data and Index Organization using the (1: m) Interleaving Technology

Efficient data placement: Chen et al. [15] has proved that different broadcasting sequence of different data would affect average access latency of data retrieval, and proposed ORD algorithm to reduce average access latency of data retrieval. Jianting and Le Gruebwald [14] proposed to reduce access latency by arrangement of the sequence of broadcast data according to retrieval frequency. Currently broadcast index studies focus on one single step to enhance efficiency without considering improving the efficiencies of the three steps. This paper has designed a new broadcast index to handle RNN query in broadcasting with considerations for the three steps.

4 A NEW INDEX FOR RNN QUERY

A RNN query that searches for q returns a collection of objects of nearest neighbors in relation to q. If we may know the distance between every object and its NN in advance, then all we have to do is to find out the distance between q and the objects which are closer than that between the objects and its NN, and then the objects are the results for the RNN query that searches for q.

The difference between Rdnn-tree and R-tree is that Rdnn-tree stores the information of every object (such as distance of neighbor, or DNN), and it may directly determine whether a leaf node is the result of the query, while R-tree cannot directly determine whether a leaf node is the result of the query and must use branch-and-bound technique, which may cause back-tracking problem. Therefore, we further improve Rdnn-tree with the principles for a better broadcast index that we have proposed to make it an index structure that can effectively support the RNN search in wireless broadcasting environment.

4.1 Rdnn-Tree

R-tree [2] in the early stage was an index structure developed for spatial database, and was modified to Rdnn-Tree by Yang and Lin [5] to accelerate NN and RNN queries. Rdnn-Tree structure, as shown in Figure 5, groups objects that share similar coordinates and places them on leaf node. That is, objects with similar coordinates are grouped. Then, a group of objects is contained in a smallest rectangle, which is called minimum bounding rectangle, “MBR” for short. Next, similar MBRs are further grouped; a group of MBR is contained in one even larger MBR, and the process continues until all objects are contained in the same MBR. What is stored on the internal node within an Rdnn-Tree is MBR; all nodes under it will be contained by it, and all objects will be contained by the root of Rdnn-Tree Every MBR will record the coordinate at its bottom left (M_l , M_d) and upper right (M_r , M_u), and the size and scope of a MBR can be obtained. Ptid and dnn are stored at the leaf node. Ptid is the reference number of data collection point, while dnn is the distance between the object and its NN. Ptr, rect, and MaxDnn are stored on non-leaf node. Ptr points to the address of child node, Rect is the MBR contained in the node and the child nodes underneath, and MaxDnn is the maximum value of the dnn of all objects in this sub-tree; the greatest distance between all objects and their NN under the sub-tree will not be greater than MaxDnn.

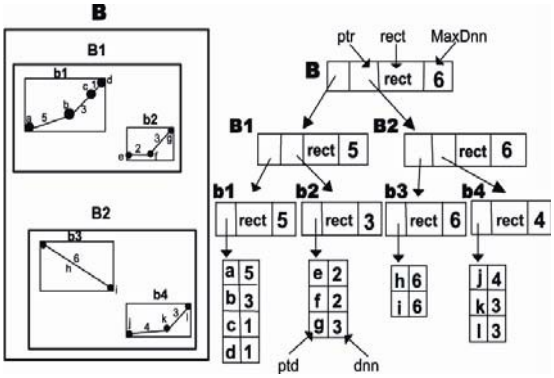


Figure 5: Data structure of Rdnn-Tree

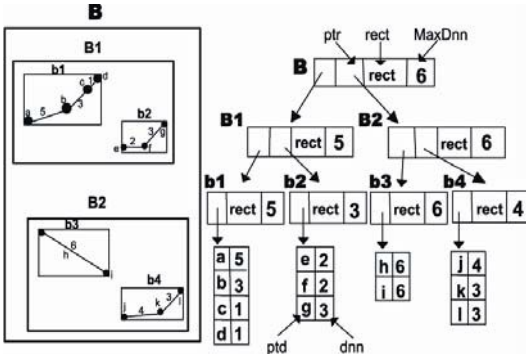


Figure 6: Data structure of Rdnn-tree

4.2 Jump Rdnn-tree

The design of a good broadcast index as mentioned in Section 3 includes three steps: reducing initial probe time, reducing size of index data, and effective placement of broadcast data. We shall explain how we improve Rdnn-tree with these three steps.

Reduction of initial probe time: The traditional approach is to increase the possibility of the appearance of index by duplicating index. However, this approach will cause longer broadcast cycle and longer average data access latency. Our approach is to build a Jump Rdnn-tree with our index structure for broadcast data. Data and index will be mixed together and broadcasted based on every sub-tree. After the index of such sub-tree has been broadcasted, the data under the sub-tree will be broadcasted in order to reduce the distance between data and index instead of broadcasting all data after the index broadcasting is completed.

Taking Figure 6 as example, the broadcasting sequence is B, B1, b1, a, b, c, d, b2, e, f, g, B2, b3, h, i, b4, j, k, and l. The cyclic index structure is also adopted. The location of each index node to be visited next is calculated with Depth First Search (DFS) traversal according to depth priority. This approach is also called jump pointer, as shown in Figure 7. Because every index node has a jump pointer, index trees are interlined through jump pointers. Therefore, index tree search may begin from any index node instead of root node. In the same time, index tree search follows pointer sequence, and there will be no back-tracking problem.

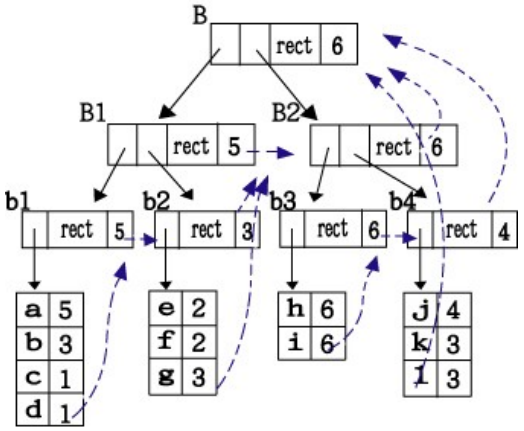


Figure 7: Data Structure of Jump Rdnn-tree

Reduction of index data size: As mentioned earlier, we adopt cyclic index structure, replication of index data for the entire broadcast cycle is not required and only one copy of index data is needed. Therefore, the size of index data is very small. Also, if the largest sub-tree at every layer of the structure of a traditional index tree has f fan-out that represents the index tree needs f pointers are needed to record the address of every sub-tree. Because of linear access feature of wireless broadcasting environment, MU can only access data in active mode, or skip the current data in doze mode and wait for the next data access in active mode. That is, only two behaviors are available: Next, which proceeds to next action, and Jump, which skips data retrieving. If the criteria for Jump are not met, then Next takes place and Next behavior does not have to be recorded. Therefore, any search for non-leaf node of a tree requires only keeping one jump pointer, and other f-1 pointers for sub-trees can be eliminated, leading to reduction of index data size.

Effective data placement: Query point is produced based on the entire search space. The possibility of the occurrence in every area shall affect searching efficiency. If the corresponding index data for the

area where the possibility of the occurrence of query point is higher are broadcasted earlier, then the index search efficiency will be better and if the location of the query point is in uniform distribution. Therefore, the broadcast sequence of a sub-tree is determined according to the area of MBR where the sub-tree is, because the larger the area of MBR is, the higher the possibility of the query occurrence will be.

After Rdnn-tree is improved by the above-mentioned approach, the problem of back-tracking is eliminated, and Rdnn index tree with cyclic structure fits with linear access feature of broadcast better. The detailed algorithm of RNN queries of cyclic broadcast is illustrated in Algorithm 1: RNN-Search-On-Air. D (q, ptid) represents the distance between query point q and objects ptid, wile D (q, rect) represents the distance between query point q and rectangle rect.

RNN-Search-On-Air (Node n, Point q)

Case 1: If n is leaf node then

For all (data-item, dnn) in n

If D (q, ptid) < dnn, then data-item is the RNN for q.

Case 2: If n is non-leaf node, then

For all branch B = (ptr, MBR, Maxdnn) in n
If D (q, rect) < MaxDnn, then call RNN-Search-On-Air (B.ptr, q).

Algorithm 1: RNN Search on Air

5 EXPERIMENT ENVIRONMENTS

5.1 Two Different Experiment Data Sets

We use two different data sets for the experiment as shown in Figure 8. For the first dataset, UNIFORM, we produce 1,000 points in square Euclidean space uniformly. For the second data set, SKEW, we produce 1,000 points with Zipf distribution, and the skewness parameter of Zipf distribution is 1.2.

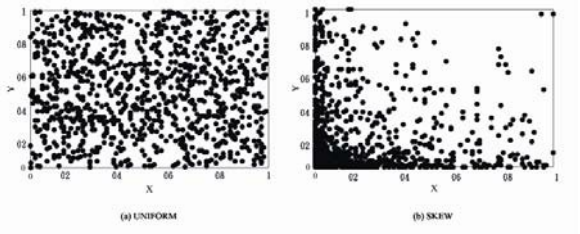


Figure 8: Uniform and Skew data sets

5.2 Compared Algorithm

Due to the feature of linear access of wireless broadcasting, and therefore Rdnn-Tree is modified to fit in the air indexing model. In a node of Rdnn-Tree, it will access to data by DFS sequentially according to depth searching. The sub-tree branches that do not match with the condition of distance heuristics search in the process of searching will be pruned. In order to reduce access latency, the ratio of Rdnn-tree and sorted list of the broadcast index here is set to 1:m. Interlace techniques are called Rdnn-Tree (1: m).

5.3 Performance Metrics

The metrics used for measuring the effectiveness of On Air Index are access time and tuning time, and the unit of measurement is packet (the unit of broadcast). Although tuning time in general may reflect the power consumption of mobile devices, but it only records the power consumed on active mode; therefore, it may not reveal the actual power consumption. Although less power is consumed in doze mode, but as the waiting time is prolonged, the power consumption is also very huge. We believe power conservation should be evaluated with total power consumption; therefore, total power energy metric is added to the performance metrics used in our experiment. Total power energy is $P=1200*Time_{active\ mode} + 60*Time_{doze\ mode}$. In order to simplify the complexity of this experiment, we ignore the power consumed in query processing under the premise that the result of the experiment is not affected. Assume 1200mW includes the power required for accessing one unit of broadcast packet, 60mW is the power required for waiting for one unit of broadcast packet.

5.4 Parameters Setting

Parameters setting are shown in Table 1. The packet id of each packet is 2 bytes; one coordinate is 4 bytes, and index takes up 2 bytes. Packet size varies from 64 bytes to 1024 bytes. Fan out of Jump Rdnn-Tree is set to 6. The number (object #) of the parameter object varies from 1000 to 5000. Query is randomly produced from the entire search space. User's initial probe time is randomly produced from 1 to 5000 broadcast unit. The final statistic result is an average value of 30 queries [19]. The program used for the experiment is modified with the R-tree codes of R-Tree Portal (http://www.rtreeportal.org/).

Table 1: Experiment Metrics Setting

Parameter	Description	Setting
Object#	number of data object	default:1000 vary from 1000 to 5000
Packet Size	size of a broadcast packet	default:256 bytes vary from 64 to 1024 bytes
Fan out	number of the sub-trees of Rdnn-Tree	default t:6

6 PERFORMANCES RESULTS

6.1 Influence of Packet Size on Performance

This experiment is to measure the performance efficiency for different packet size under two different data sets. Experiment results of different data are shown in Figure 9 and Figure 10.

For access time, no matter data object is in uniform distribution or skew distribution, access time becomes smaller as the size of broadcast packet changes, because larger packet capacity of broadcast packet allows more data. Therefore, the broadcast cycle of entire broadcast program will be shorter. Besides, our approach is obviously better than Rdnn-tree (1:m). In our approach, index data is broadcasted only once. Compared to Rdnn-tree that adopts (1: m) and broadcasts index data m times, our approach has relatively shorter broadcast program cycle and less access latency.

For tuning time, as broadcast packet capacity increases, the average packet access time decreases. However, when object data is in skew distribution, if broadcast packet capacity is larger than 1024 bytes, then tuning time will increase slightly. We adopt the approach in which data is mixed with index and broadcasted, and the entire index data in DFS sequence is scattered in the entire broadcast program; when the broadcast packet capacity is large enough, index sections may not fill a broadcast packet to the fullest. Because a broadcast program with index data must separate index and data packet, a broadcast packet not fully loaded still occupies one broadcast packet. This situation tends to be more severe when the object data is in skew distribution. Therefore, when broadcast packet capacity becomes larger, the entire average broadcast packet access time decreases, and when average packet access time is shorter, its efficiency is more significant.

For total power consumption, regardless of data object distribution, our approach is significantly better than Rdnn-Tree (1: m). Even though the tuning time in our approach is slightly larger than Rdnn-Tree (1: m) when object data is in skew distribution; the broadcast packet capacity is larger than 1024 bytes, the total power consumption in our approach is still better. This matches with our idea. When considering broadcast index efficiency, total power consumption must be also considered because it reveals the real power consumption of mobile devices.

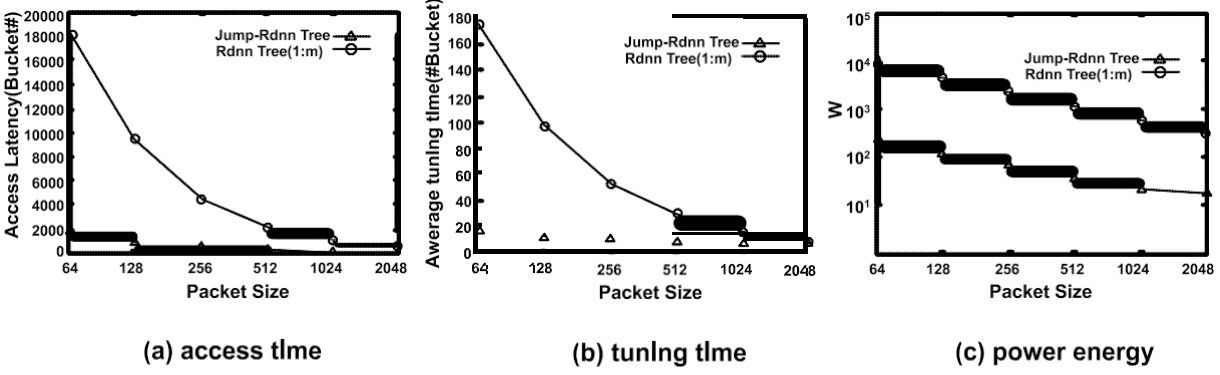
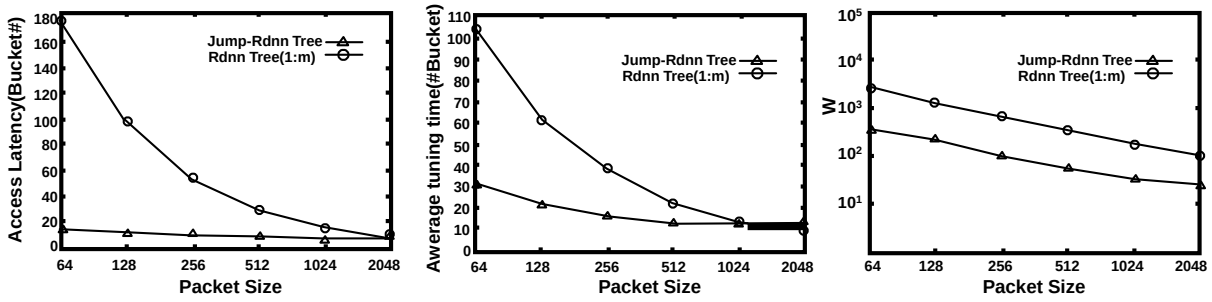


Figure 9: Influence of packet size on performance when data objects are in normal distribution



(a) access time **(b) tuning time** **(c) power energy**

Figure 10: Influence of packet size on performance when data objects are in skew distribution

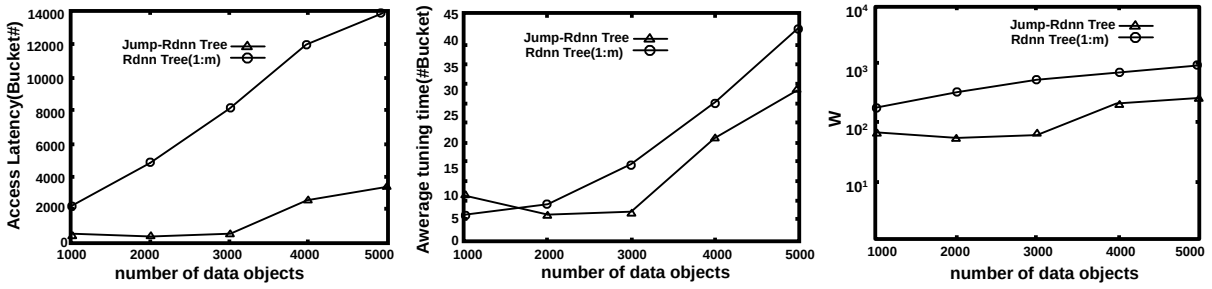
6.2 Effect of Data Object on Experiment Effectiveness

This experiment is to measure the performance efficiency for different number of data objects (object#) under two different data sets. Experiment results of different data are shown in Figure 11 and Figure 12. As the number of data objects increases, broadcast program cycle lasts longer, making access latency, tuning time, and total power consumption increase.

For access latency, because the entire broadcast program cycle is longer, average waiting time for packet broadcast is longer, and the size of index data is larger, the performance ratio of Rdnn-Tree (1: m) is m times.

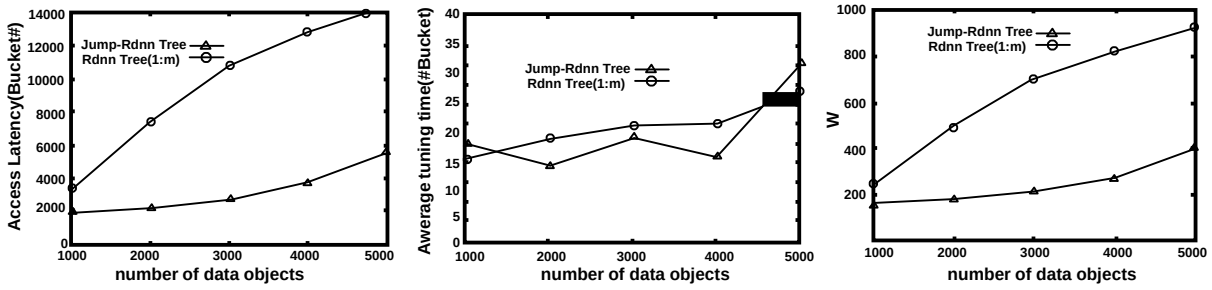
For tuning time, as mentioned earlier, because a user turns into active mode for data retrieval or doze mode for skipping the retrieval according to broadcast data, the capture of selective tuning data must separate index packet and data packet.

For total power consumption, our approach is significantly better than Rdnn-Tree (1: m).



(a) access time **(b) tuning time** **(c) power energy**

Figure 11: Effect of number of data object on performance when data objects are in normal distribution



(a) access time **(b) tuning time** **(c) power energy**

Figure 12: Effect of number of data object on performance when data objects are in skewed distribution

7 CONCLUSIONS AND FUTURE WORK

In this work we have discussed how to effectively organize and deploy data in wireless broadcasting environment and answered questions about RNN query. Based on previous studies, we have summarized the principles for designing broadcast index. Based on these principles, a new index structure, Jump-Rdnn tree that is idea for RNN query in broadcast environment, is designed. Because this work focuses on location-dependent data access in broadcasting environment and is different from traditional on-demand access model, we only discuss issues concerning static RNN query. We shall further extend to more advanced and more dynamic location-dependent queries.

8 REFERENCES

[1] A. Datta, D.E. Vandermeer, A. Celik, and V. Kumar,"Broadcast Protocols to Support Efficient Retrieval from Database by Mobile Users", ACM Transactions on Database Survey

[2] A. Guttman , "R-Trees: A Dynamic Index Structure for Spatial Searching" , Proceedings of the 1984 ACM SIGMOD international conference on Management of data , 1984 , p.p.47-57.

[3] B. Zheng , W.C. Lee and D.L. Lee, "Search K Nearest Neighbors on Air", Proceedings of the 4th International Conference on Mobile Data Management, Melbourne, Australia January 2003,p.p. 181-195.

[4] B. Zheng and D.L. Lee,"Information Dissemination via Wireless Broadcast", Communication of ACM, 48(5), May 2005, p.p. 105-110.

[5] C. Yang and K.L. Lin, "An index structure for efficient reverse nearest neighbor queries", Proceedings of the 17th International Conference on Data Engineering,2001,p.p. 485-492.

[6] Computer Science and Telecommunication Board. IT Roadmap to a Geospatial Future, the National Academies Press, 2003.

[7] D. Barbara, "Mobile Computing And Databases - A Survey", IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 1, February 1999 , p.p. 108-117.

[8] D.L. Lee, W.C. Lee, J. Xu and B. Zheng, "Data Management in location dependent services", IEEE Pervasive Computing, Vol. 1, No. 3, September 2002, p.p. 65-72.

[9] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries ", Proceedings of the ACM SIGMOD International Conference on Management of Data, ,May 16-18, 2000 , p.p. 201-212.

[10] I. Stanoi , D. Agrawal and A.E. Abbadi, "Reverse Nearest Neighbor Queries for Dynamic Databases", ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000, p.p.44-53.

[11] J. Xu , B. Zheng , W.C.Lee and D.L. Lee", "Energy efficient index for energy query location-dependent data in mobile environments", In Proceedings of the 19th IEEE International Conference on Data Engineering, Bangalore, India March 2003, p.p. 239-250.

[12] J. Zhang, M. Zhu ,D. Papadias ,Y. Tao and D.L. Lee, "Location-based Spatial Queries", Proceedings of the 2003 ACM SIGMOD international conference on Management of data, San Diego, California, USA , June 9-12 2003,p.p. 443-454.

[13] J. Xu and W.C. Lee and X. Tang ,"Exponential Index: A Parameterized Distributed Indexing Scheme for Data on Air"., "Proceedings of the 2nd ACM/USENIX International Conference on Mobile Systems, Applications, and Services , Boston, MA, June 2004,p.p. 153-164.

[14] J. Zheng and L. Gruenwald, "Prioritized Sequencing for Efficient Query on Broadcast Geographical Information in Mobile Computing", ACM GIS, 2002,p.p.88-93.

[15] M.S. Chen ,P.S. Yu and K.L. Wu, "Optimizing index allocation for sequential data broadcasting in wireless mobile computing", IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 1,January-February 2003,, p.p. 161-173.

[16] M. A. Viredaz, L. S. Brakmo and W. R. Hamburgen, "Energy management on handheld devices", ACM Queue, Vol. 1, No. 7, October 2003, p.p. 44-52.

[17] N. Roussopoulos, S. Kelley and Fr'ed'eric Vincent, Nearest neighbor queries, Proceedings of ACM SIGMOD International Conference on Management of Data, June 1995, p.p. 71-79.

[18] Q. Hu , W.C. Lee and D.L. Lee, "Index Techniques for Power Management in Multi-Attribute Data Broadcast", Mobile Networks and Applications, Vo. 6,No. 2,2001,p.p.185-197.

[19] Sheldon Ross, Introduction to Probability and

Statistics for Engineers and Scientists, 2000.

[20] T. Imielinski ,S. Viswanathan and B. R. Badrinath, Data on Air:organization and access, IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 3, May 1997, p.p. 353-372.

[21]R. Benetis, C.S. Jensen, G. Karciauskas and S. Saltenis, Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects, International Database Engineering and Applications Symposium, Canada, July 17-19, 2002, p.p. 44-53.

[22] G. Iwerks, H. Samet and K. Smith, "Continuous K-Nearest Neighbor Queries for Continuously Moving Points with Updates", International Conference on Very Large Data Bases, 2003, p.p. 512-523.

[23] Z. Song and Nick Roussopoulos, K-Nearest Neighbor Search for Moving Query Point, Proceedings of 7th International Symposium on Advances in Spatial and Temporal Databases, LNCS2121, Redondo Beach, CA, USA, July 12-15,2001,p.p. 79-96.