Scalable Computing: Practice and Experience
Volume 8, Number 3, pp. 291–300. http://www.scpe.org
ISSN 1895-1767

DHR-TREES: A DISTRIBUTED MULTIDIMENSIONAL INDEXING STRUCTURE FOR P2P SYSTEMS

XINFA WEI AND KAORU SEZAKI*

Abstract. Supporting range query over Peer-to-Peer systems has attracted many research efforts in recent years. In this paper, we propose a new multidimensional indexing structure for P2P systems called Distributed Hilbert R-trees (DHR-trees). DHR-trees enables multidimensional range query to be executed similarly as in overlapping regions tree in P2P systems. Its distributed structure makes it fault-tolerant and scalable to dynamic network environment with a large number of peers as well. Our experiments shows that it performs well on multidimensional range query while the maintenance cost is reasonably low.

Key words. peer-to-peer systems, multidimensional queries, Hilbert rtrees

1. Introduction. Enabling efficient access to distributed and dynamic data is hot research topic in P2P systems. Complex queries, such as range queries and k-nearest neighbors queries over multidimensional data, are becoming more important as large amount of data are shared between thousands of user applications. For example, a P2P auction network [17] where peers store information on local real estate (geographical location, price, etc) needs to frequently deal with queries such as "find all real-estate advertisement for properties in a given region in a city". Another query example is that P2P-based sensor data provision networks for environmental surveillance needs to be able to answer request such as "find the average temperature and dioxide concentration within 100 meters from point A".

Multidimensional indexing and range queries problem has been extensively studied in world of the databases. However, in P2P context, this problem brings more challenges because it requires both efficient network-based query processing, fault-tolerant network topology and low cost of network maintenance. Without any central index server, all shared data must be reachable and searchable with the network constantly changing. Each peer node, after receiving query request, should be able to initiate query and decompose query into some sub-queries which are forwarded to other peer nodes when necessary. Therefore one peer node should keep some links and data distribution information to some other peer nodes in the network. There are two extreme solutions for this problem. One is to let peer node keep nothing about information of other nodes and always flood query across the network. This preliminary method is used in earlier P2P systems and is obviously not practical as the network grows very large. Another extreme is to let every peer node keep and refresh all other nodes information all the time to enable query to be executed efficiently and accurately. But maintaining such information requires huge sum of exchange messages which is not affordable in the environment where changes occur frequently in large network.

In this paper, we propose a decentralized multidimensional indexing structure, called DHR-trees (Distributed Hilbert R-trees), that is capable of supporting range queries efficiently and doing network maintenance at a low cost. In DHR-trees, each peer is assumed to control a data set within a minimum bounding rectangle in the space and each peer can be regarded as a leaf node in a Hilbert R-tree. DHR-trees preserves the feature of overlapping regions techniques as in traditional centralized Hilbert R-trees (or R-trees [8] and its variants) which make it can easily accommodate existing R-trees based algorithms with minor modification. DHR-tree of order d provides $O(\log_d N)$ search cost for equality queries, where N is the number of peer in the system. In addition, we introduce the region maintenance of DHR-trees that addresses dynamism problem of P2P systems where peers may be inserted/deleted frequently. We have done some evaluation test of DHR-trees system under various settings. Experiments are performed with different network size, different types of data distribution etc. By testing various settings, the results proved the correctness of P2P systems and effectiveness of range query algorithms. Comparison results also demonstrates that its query execution requires much less node visits than in Squid [14] P2P system.

The rest of this paper is organized as follows: in Section 2, related works are introduced. We then describe in Section 3 DHR-trees structures and the implementation issues. In Section 4, simulation results and comparison result are shown. Finally, conclusion is presented in Section 5.

*The University of Tokyo, 4-6-1 Komaba, Meguro, Tokyo, 153-8505 Japan, email: wei@mcl.iis.u-tokyo.ac.jp, sezaki@iis.u-tokyo.ac.jp

2. Related Work. Many P2P systems have emerged in recent years. Earlier ones are mainly for file sharing applications such as Gnutella [1] and Kazza [2]. They search requested files by flooding messages across network. Unbounded cost causes heavy traffic overhead. Later structured P2P systems, such as Chord [16], Pastry [13], CAN [12] and Tapestry can efficiently support key-based routing and equality searching with logarithmic routing cost. However, the hash function used by these systems destroys the order in the key value space, making them impractical for processing range queries. Recent systems, such as P-Tree [5], P-Grid [3], Skip Graphs [4] and BATON [9], can support one dimensional range queries.

Application with multidimensional data are very important in practical world. Aforementioned systems can not deal with multidimensional data. In contrast, there are some multidimensional index structure solutions available in the centralized database world. However, they are not directly usable in P2P context. Without any central index server, the highly dynamic P2P system must ensure that query can be always answered without missing data. Letting each node maintain links to all other nodes will lead to scalability problem. Ganesan et al. [7] concluded that adapting these solutions to the P2P systems presents four challenges: distribution, dynamism, data evolution and decentralization. Some research efforts has been made and there are progresses in supporting multidimensional range queries. We pick up some most related works and list them as follows:

Mondal et al. [11] proposed P2PR-trees, a variant of R-trees that targets P2P networks. They showed in their simulation study that P2PR-trees show better scalability than two-level Master Client R-trees, since they do not suffer from the central server bottleneck. However, the most important issue in P2P systems, maintenance of a dynamic R-tree is left unaddressed.

The ZNet [15] partitions the data in the native data space in a way as in the generalized quad-tree. ZNet makes use of the Skip Graphs [4] as overlay network routeing in one dimensional space, therefore multidimensional data space is mapped to an one-dimensional index space, such that it can be mapped to nodes in the network. ZNet supports range queries, however, they only provide probabilistic guarantees even when the index is fully consistent due to the underlying Skip Graphs. Moveover, the search performance of the ZNet is $O(d \cdot \log_d N)$ while search performance of DHR-trees is $O(\log_d N)$.

Squid's proposed structure is the closest one to DHR-Trees. In [14], Schmidt C. et al. presents the design of Squid P2P System and its evaluation. The space is first mapped down into an one dimensional space using a Hilbert space filling curve. The one dimensional data is then range partitioned in one dimension and mapped onto the Chord [16] overlay network. For load balancing purpose, however, the original Chord protocol requires that data identifiers are uniformly distributed into one-dimensional space. This restriction makes it inappropriate to directly use location as a data identifier, because data can not always be guaranteed of being distributed uniformly in the space. Furthermore, since routing relies on Chord, which is originally designed for equality search only, the query type is also limited. For example, realizing nearest neighbor query is not easy with Squid since there is no spatial information being preserved.

DHR-Trees also makes use of Hilbert curve to map multiple attributes from m-dimension down into one-dimensional space. But the spatial overlapping region information is kept by individual peers and dynamically maintained. This facilitates multidimensional spatial queries in being executed more efficiently. Unlike other similar approaches, such as Squid and SCRAP [7], DHR-Trees distinguishes itself in supporting R-Tree-like complex queries in dynamic P2P networks.

3. DHR-Trees P2P structure. In multidimensional database, R-Tree [8] and its variants use tree structure for storing information. Each node of an R-tree has a variable number of entries (up to some pre-defined maximum). Each entry within a non-leaf node stores two pieces of data: the child node identifier, and the bounding rectangle of all entries within this child node. Thus, a R-Tree can be regarded as an overlapping region (i. e. rectangle) tree. Search algorithms are usually run from root of the tree; using bounding rectangles to decide whether or not to search inside a child node. Therefore, most nodes in the R-Tree are never "visited" during a search. This leads to high efficiency on query execution.

The Hilbert R-Trees, one of the best-performing multidimensional index structures [6] in the R-Trees family, combines the overlapping regions technique of R-Tree with Hilbert space filling curves. It first stores the Hilbert values of the data rectangle centroid in a B+-tree, then enhances each interior B+-tree node with the minimum bounding rectangle of the sub-tree below. This facilitates the insertion and deletion of new objects considerably. Figure 3.1 demonstrates two dimensional space with some data objects and Figure 3.2 shows its corresponding Hilbert R-Tree.

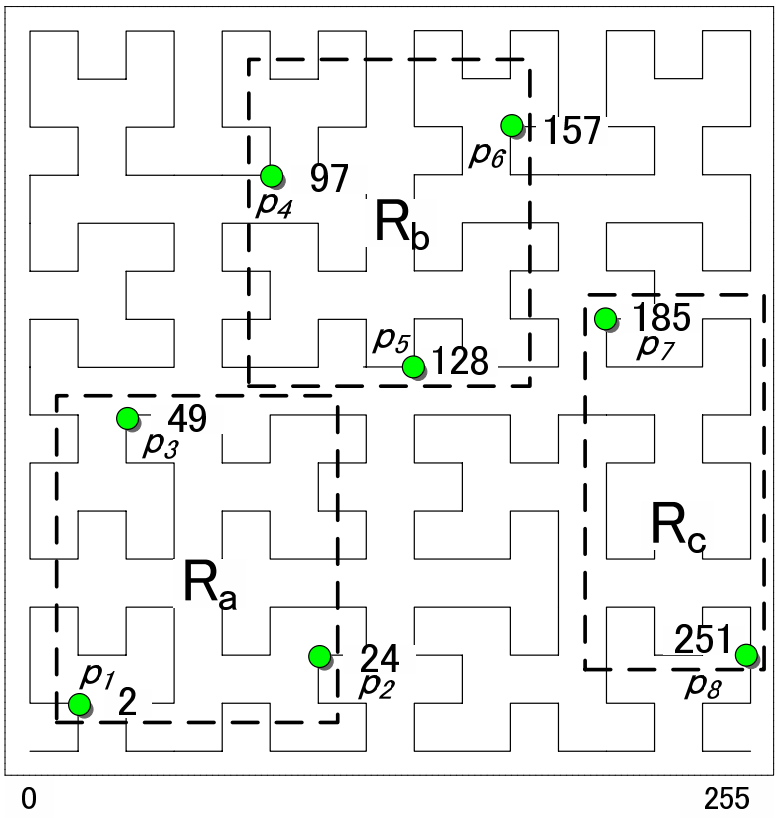


FIG. 3.1. *Data rectangles in a Hilbert R-Tree*

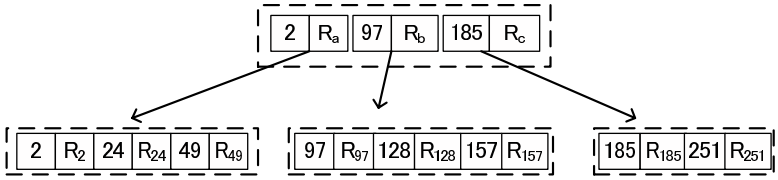


FIG. 3.2. *A Hilbert R-Tree*

P-Trees [5], a recently proposed Peer-to-Peer system, enables one-dimensional range queries to be executed collaboratively by peers. Instead of using Distributed Hash Tables method, it organize all nodes into a closed virtual ring by node identifiers (without hashing). As a result, one-dimensional range queries becomes possible, since the proximity between peers' identifiers is preserved. Each peer independently maintains a routing table, where entries are exactly those on the left-most root-to-leaf path of the B+-Tree. Each peer has its own view of B+-Tree, in which the peer's identifier is located at the left-most leaf node. To keep freshness and correctness of the routing table, peers in the P-Trees system periodically call a stabilization method (called stabilizeLevel).

DHR-Trees combines the advantages of P-Trees and Hilbert R-Trees together. It takes the same topology as in P-Trees. It also enhances peer's routing table with bounding rectangle information. It can be loosely regarded as a distributed version of Hilbert R-Trees, but each node has independent view of the a Hilbert R-Tree. It can also be regarded as a multidimensional extension of P-Trees, enabling indexing and flexibly searching multidimensional information in P2P systems.

3.1. Overview of DHR-Trees. The idea is partially inspired by and based on the work of P-Trees [5], which is designed for supporting one-dimensional range queries in P2P network. The main enhancement is that each peer keeps relevant region tree information in its routing table. Hence it supports the same class of queries as in centralized R-Trees. This is significant since many complex multidimensional query algorithms

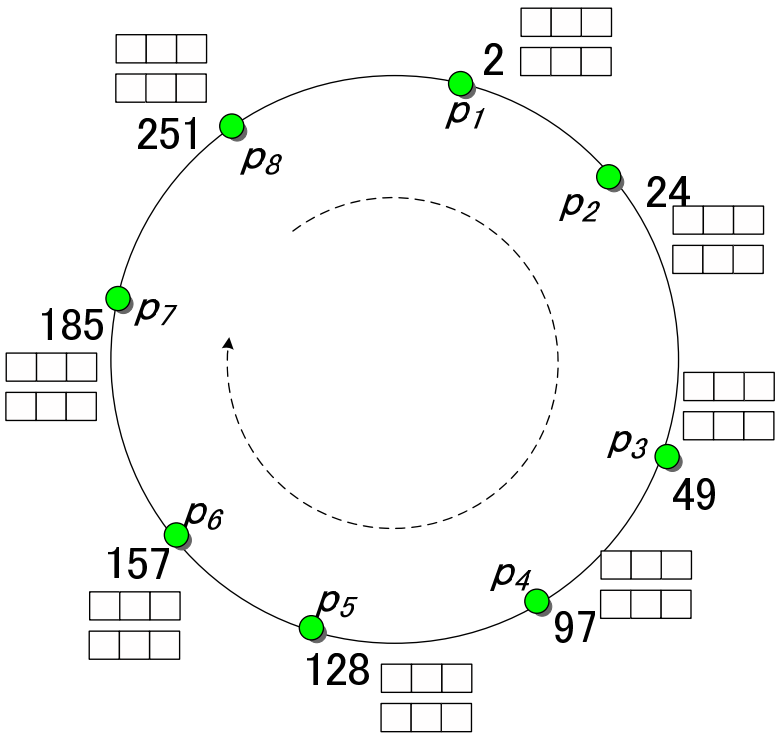


FIG. 3.3. *Virtual ring of peers with routing tables*

can be adapted into the P2P network environment. It discards the notion of maintaining and sharing a globally consistent R-Tree by all peers, and instead maintains semi-independent DHR-Trees at each peer. This allows for fully distributed index maintenance without any need for inherently centralized and unscalable techniques such as primary copy replication.

In a DHR-Trees P2P system, each peer is assumed to control some data set comprised by a Minimum Bounding Rectangle (MBR) in the multidimensional space. Peer p is associated with data rectangle $p.MBR$ and with $p.HCode$, where $p.HCode$ is the Hilbert value of centroid of $p.MBR$ from the space. The $p.HCode$ is used as the index value (key) of p in the one-dimensional space. At the underlying network layer, each peer identifies itself by $p.NetAddress$, which is usually the IP network address. Peers organize themselves into a virtual ring as in P-Tree by their $p.HCode$. The Hilbert curve therefore becomes a closed ring and peers are indexed by their $p.HCode$ in the ring.

Figure 3.3 demonstrates the structure of DHR-Trees for the two-dimensional space case as shown in Figure 3.1. For simplicity, each is assumed to store one point data at its location. By mapping from two-dimensional to one-dimensional space, each peer has a unique Hilbert value. These values are used as peer identifiers in the DHR-Trees P2P system. Peers organize themselves into a virtual ring by their identifiers. Each peer has a pair composed of the predecessor peer and the successor peer as in Chord [16] system. Each peer also has its own composite routing table. The composite routing table is the most important component of DHR-Trees. With Hilbert value and coverage information in the routing table, DHR-Trees P2P system supports routing and equality search similarly as in P-Trees. By having spatial region information in the composite routing table, DHR-Trees P2P system supports multidimensional query directly and efficiently as in centralized R-Trees.

3.2. Composite Routing Table. In DHR-Trees systems, each peer stores nodes information of the left-most root-to-leaf path of independent Hilbert R-Trees from its view in its composite routing table. The significant enhancement to P-Tree is that the region information MBR (we inherit the word MBR from the database world for convenience) is stored as part of the multidimensional overlapping regions tree. The MBR is a minimum bounding rectangle of the sub-tree below (see example in Figure 3.4). As an important part of DHR-Trees, the MBR entry facilitates multidimensional queries, e.g. range queries could be executed efficiently as in centralized R-Trees. Details of the routing table are presented below.

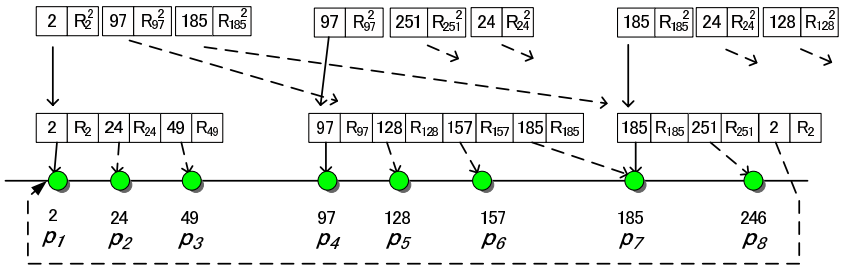


FIG. 3.4. *An example of DHR-Trees with routing tables at p_1 and related peers*

Similar to the P-Tree structure, a peer p maintains a routing table $p.node$, a double indexed array, containing nodes information. Formally, the routing table $p.node$ consists of $numLevels$ rows. There are $p.node[i].numEntries$ of node entries at level i, where $0<i \leq numLevels$. The $numLevels$ is at most $[log_d N]$, where N is the number of peers being indexed and d is the order of the R-Tree. The $p.node[i].numEntries$ is between d and $2d$ at the non-root level of the tree. At root level, $p.node[i].numEntries$ is between 2 and $2d$. Each entry in the row is a triple as following:

$$\langle HCode, MBR, NetAddress \rangle$$

which points to the peer *peer* that owns the MBR and is identified by the index value $HCode$. In the routing table $p.node$, every *peer*'s MBR at level i is a minimum bounding rectangle that comprises all $peer.node[i-1][j].MBR$, where $1 \leq j \leq peer.node[i-1].numEntries$.

Figure 3.4 shows the composite routing table of peer p_1 and related peer for intuitive purpose. The number of levels of p_1, the $p_1.numLevels$, is 2. There are three entries at each level where the first one is about p_1 itself. At level 1, there are 3 entries corresponding to the triples (2,R_2,p_1), (24,R_{24},p_2) and (49,R_{49},p_3) (Due to space limitation, the $NetAddress$ such as p_1 and p_2 etc. is not shown). R_2 is p_1's data MBR. R_{24} and R_{49} are acquired from p_2 and p_3 respectively during p_1 stabilization process on level 1. Notice that p_1,p_2 and p_3 in the triples represent the network address of these peers respectively. At level 2, there are three entries corresponding to the triples (2,R_2^2,p_1), (97,R_{97}^2,p_4) and (185,R_{185}^2,p_7). The MBR at level above 1 is tagged with level i as superscript, such as R_2^2, R_{97}^2 and R_{185}^2. R_2^2 is calculated and updated by calling Algorithm 2. The value of R_{97}^2 and R_{185}^2 are updated at their corresponding peers, p_4 and p_7, and in turn these values are obtained by p_1 during p_1's later stabilization process on level 2. We notice level 1 consists of the closest succeeding peers. At level 2, entries are succeeding peers with *widened* intervals. The interval is equivalent to the coverage range in P-Tree. We can see that these rectangles can be regarded as the left-most route-to-leaf part of the "sliding" Hilbert R-Trees, which has p_1 as its left-most leaf node.

Algorithm 2: p.UpdateMBR (int i)

```
Input: i level of route table, i >1
begin
    reset p.node[i][1].MBR to empty
    j = 1
    while j <p.node[i-1].numEntries do
        p.node[i][1].MBR = CombineRect(p.node[i][1].MBR, p.node[i − 1][j].MBR)
        j++
end
```

3.3. Range Queries. From introduction in previous subsection, it is clear that the rectangle-based overlapping regions tree is maintained in a distributed fashion among peers. This characteristic makes DHR-trees capable of doing R-tree-like range query in P2P systems. Due to the similarity of point query and range query, we focus on how range query is executed.

THEOREM 3.1. *Given the DHR-trees in consistency state, range query initiated at any peer can finally be answered with all qualified data object that exist in the network with guarantee.*

Algorithm 3: p.WindowQuery (Rectangle w, int l)

```
Input: w query window
Input: l level of search to start
begin
    j = 1
    localIntersect = false
    if l >1 then
        while j <p.node[l].numEntries do
            if p.node[i][j].MBR intersects w and l >1 then
                p' = p.node[i][j].peer
                p'.WindowQuery(w, l − 1)
                if j = 1 then
                    localIntersect = true
            j++
    if l = 1 then
        while j <p.node[l].numEntries do
            if p.node[i][j].MBR intersects w then
                p' = p.node[i][j].peer
                p'.searchLocalData(w)
            j++
    if localIntersect then
        this.WindowQuery(w, l − 1)
end
```

In DHR-trees, each peer is ordered by a one dimensional value - Hilbert code and all peers are orderly indexed in the ring of underlying P-Tree. When the P-Tree reached consistent state, any two neighboring entries at same level in route table are guaranteed by P-Tree with coverage property, that is, no peer "missed" by the index structure. Therefore, the coverage property ensures that all peer entries at lower level are contained by entries at higher level. Since each MBR is a combination of all lower *peer.MBR* and the $UpdateMBR$ following level stabilization that works bottom-up from the lowest level to highest level, we can prove all MBR at higher level also comprise all lower MBR by induction bottom-up. Therefore any *peer.MBR* is contained by root level entries. As a result, DHR-tree can guarantee all qualified data object for a query being returned. The pseudo-code of range query process is listed in Algorithm 3. Notice that the result, usually being composed of several pieces, is returned in a asynchronous fashion due to different path length and network latency. The peer initiated query has to be responsible for collecting and synthesizing.

3.4. Wrapping-around problem. Since all peer nodes are indexed in a Hilbert curve ring, a problem about MBR arises: when a peer is near the end of the curve, the MBR entries (i. e. *p.node[i][0].MBR*s) at the upper level of the routing table will need to cover a group of *peer.MBR* located near the start of the Hilbert curve, which is distant from *p.MBR* in the m-dimension space. Intuitively, redundant space will be possibly comprised and it will result in increasing search cost. To solve this problem, we introduce an auxiliary MBR, denoted as *AuxMBR*, for containing the wrapped-around rectangles, while MBR still contains non-wrapped ones. Therefore a more general structure of a route entry should be

$$\langle HCode, MBR, AuxMBR, NetAddress \rangle$$

A null value of *AuxMBR* is allowed when no wrapping-around MBR exists. Accordingly, the range query search (Algorithm 3) should be also altered. Experiment results illustrated in Figure 4.6 shows that the routing cost decreases when this solution is applied.

4. Evaluation. In this section, we present some experimental results we conducted experiments by a implementation of DHR-trees. All our experiment software was implemented in Java and based on PlanetSim [10].

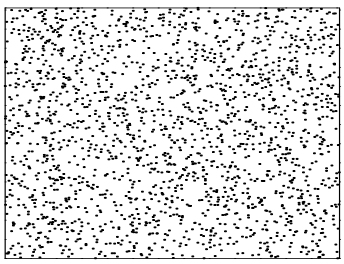


FIG. 4.1. *Uniform*

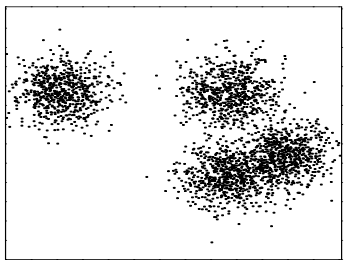


FIG. 4.2. *Clustered*

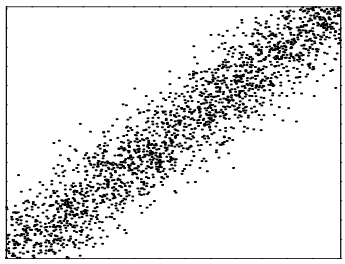


FIG. 4.3. *Skewed*

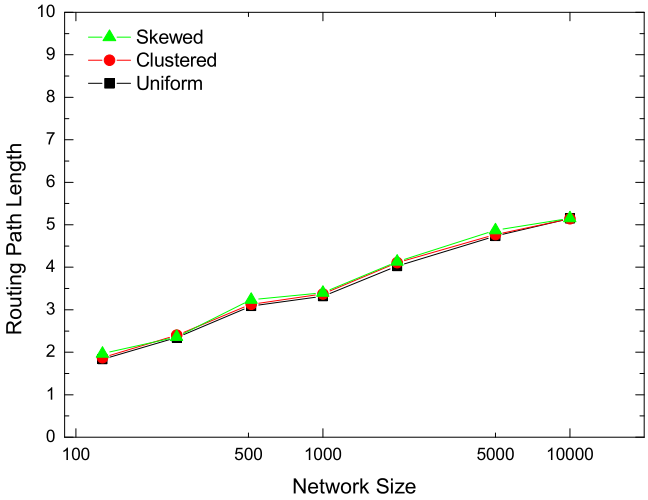


FIG. 4.4. *Routing Cost*

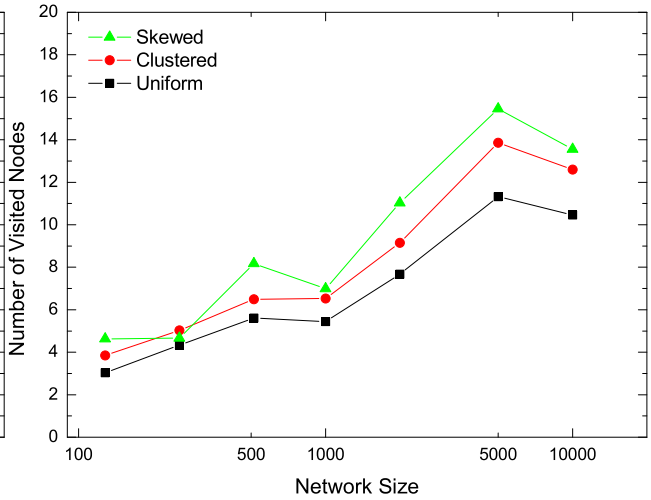


FIG. 4.5. *Point query cost*

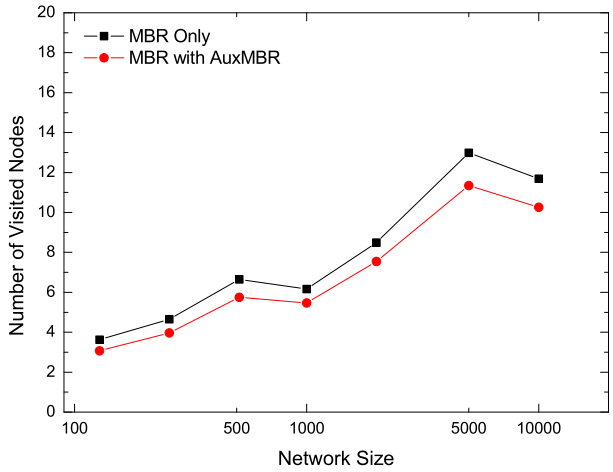


FIG. 4.6. *Improvement with Auxiliary MBR*

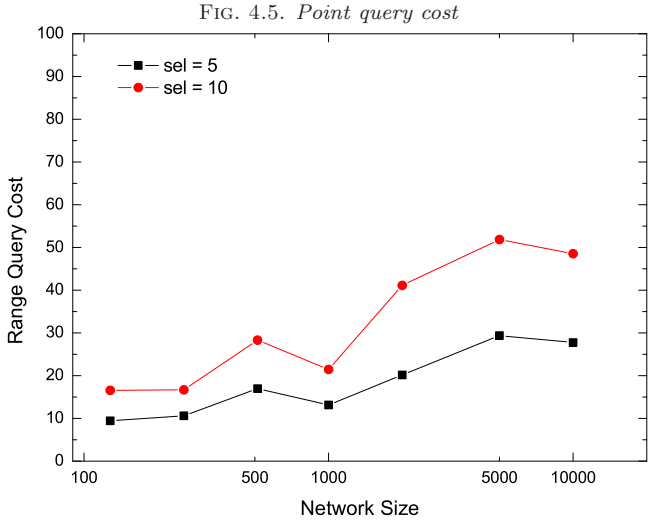
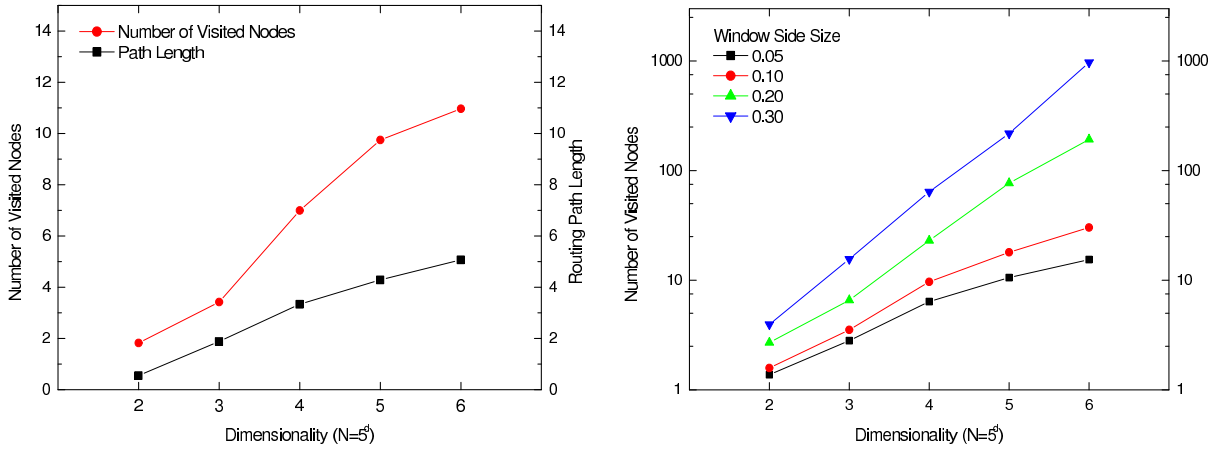


FIG. 4.7. *Range Queries*

All experiments were performed on a machine (3.0GHz Intel Pentium ET) with 2GBytes of main memory, running Windows XP Professional. In our current simulation environment , the DHR-trees nodes were configured to run in a single Java Virtual Machine.

4.1. Two-dimensional DHR-trees. In our simulations, we first use a two dimensional space. This space is mapped by a Hilbert curve of order 16 down into a one dimensional space such that it is partitioned into a $2^{16} \times 2^{16}$ grid. For simplicity of evaluation, each peer is assumed controlling a rectangle of size 1×1. The order d of R-tree is fixed as 4 and network size varies from 128 to 10000. Simulation program begins with building DHR-trees with configuration as the initial phase. Node joins into DHR-trees every several steps in a random order and stabilization process of P-Trees works periodically together with $UpdateMBR$ process.



(a) Routing path length and Query cost (b) Range queries

FIG. 4.8. *Multidimensional DHR-trees*

Insertion/deletion, routing and query experiments are executed after the network becomes stabilized. Three type of distribution of data sets are used: uniform, clustered and skewed distributions as shown in Figure 4.1, Figure 4.2, Figure 4.3 (Each with 2000 points in Figures).

Point Query Cost. Poinst query cost is measured by number of visited nodes during query. The different distribution incurs difference on number of visited nodes as shown in Figure 4.5: routing in case of skewed distribution visited more number of nodes. This is because that skewed data set make more extensive overlapping MBR in route table so that routing are more frequently forwarded to some unnecessary nodes.

Improvement by Auxiliary MBR. In Section 3.4, the auxiliary MBR is introduced to solve extensive coverage problem of wrapping-around MBR. Experiment in Figure 4.6 (uniform distribution) show that this approach reduces about $12\% \sim 13\%$ routing messages in case of uniform distribution.

Range Queries. In range query experiments, we use data set in uniform distribution, where size N varies from $128 \sim 10000$. We use rectangles as query window with side length to be adjusted in accordance with network size, such that *sel* number of result peers (MBRs) could be expected to fall into query window. In our experiments, *sel* specified as 5 and 10 separately. The location of query window is randomly selected and peers that initiate query are also randomly picked from the network. Query cost are measured by number of visited nodes. For each combination of N and *sel*, 1000 times queries are executed and results are averaged. The result shown in Figure 4.7 indicates that range query can be efficiently executed in DHR-trees.

4.2. High Dimensional DHR-trees. Some experiments with high dimensional data has also been conducted. Space dimensionality m is changed from 2 to 6. Network size is set as 5^m, changing from 25 ($m = 2$) to 15625 ($m = 6$). In all dimensionality configuration, side length of space fixed at 1024 and Hilbert curve of order 10 is used. Multidimensional data is distributed uniformly in the space. Auxiliary MBR is enabled in all experiments.

Routing Path and Number of Visited Nodes. In Figure 4.8(a), length of routing path increases slowly as d and N increases. This indicates that basic routing does not require much hops even in multidimensional spaces. The number of visited nodes, i. e. point query cost, increases with network size and is always bigger than routing path like in 2-dimensional case.

Range Queries. Three group of range query experiments has been executed. In three groups, query window size is set to be 0.05, 0.1 and 0.2 respectively. As for combination of query window size and dimensionality m, the center of query window is randomly chosen in space and query is executed 1000 times to average results. Figure 4.8(b) shows: With small windows size, query cost increases slowly. With big window size, however, the query cost increases rapidly with dimensionality and network size. This can be explained as that bigger query window overlaps much more MBRs in DHR-trees in multidimensional space and more nodes will have to be involved in query.

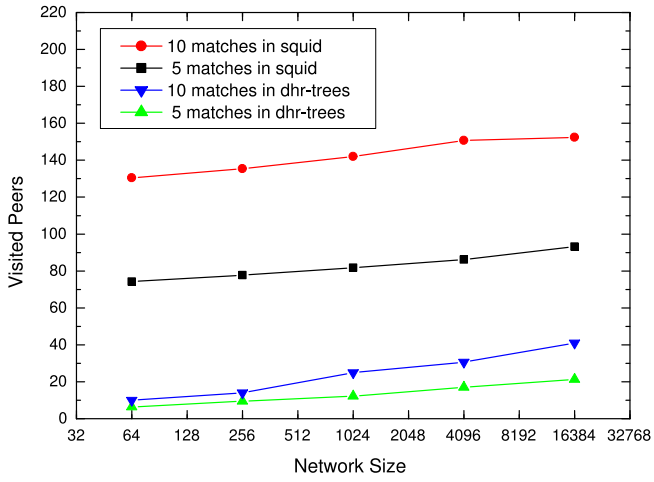


FIG. 4.9. *Query performance comparison with Squid*

4.3. Performance Comparison with Squid. As introduced in Section 2, Squid P2P system is the most related to DHR-Trees as they have the same method of mapping higher dimension space to one-dimensional space and use the same virtual ring as the overlay network structure. The most different part is that it uses a different routing table (called finger table) and a different query execution mechanism. Squid does cluster refinement to decompose query ranges into smaller query clusters recursively until it reaches *finest* level of the Hilbert curve, since top-down queries are not supported as each peer has no region information in its routing tables; In contrast, DHR-Trees supports R-Trees-like queries directly by having MBR region information maintained in peer's routing table. Hence, DHR-Trees visits less peers and generates less traffic messages when performing range queries, proving it is superior to Squid in efficiency of query execution. Figure 4.3 shows the comparison results of range queries. Moreover, nearest neighbor query is even not supported by Squid system because spatial proximity in multidimensional space is not preserved after mapping into one-dimensional space.

5. Summary. In this paper, we proposed a new multidimensional indexing structure called Distributed Hilbert R-trees. DHR-trees enables multidimensional range query to be executed efficiently as in R-trees. Its distributed structures make it fault-tolerant and scalable to a large number of peers. DHR-trees uses Hilbert curve in mapping from m-dimension down into one dimensional key space and uses P-Trees in building overlay network. Similarly as in Hilbert R-trees, this mapping is used just for ordering of each peer in the P-Tree, while inherited overlapping regions technique can efficiently support flexible queries in P2P systems. To the best of our knowledge, DHR-trees is the first P2P systems that utilizes semi-independent R-tree structure and addresses dynamic feature of network as well. Our current experiments shows that it performs well on multidimensional range query and the maintenance cost is reasonably low comparing with previous approaches. As future work, we plan to compare it with other approaches and test it in a real-world application.

REFERENCES

[1] *The gnutella web site.* `http://gnutella.wego.com`
[2] *The kazaa web site.* `http://www.kazaa.com`
[3] K. ABERER, *P-Grid: A Self-Organizing Access Structure for P2P Information Systems*, vol. 2172, Jan. 2001.
[4] J. ASPNES AND G. SHAH, *Skip graphs*, in SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, 2003, Society for Industrial and Applied Mathematics, pp. 384–393.
[5] A. CRAINICEANU, P. LINGA, J. GEHRKE, AND J. SHANMUGASUNDARAM, *Querying peer-to-peer networks using p-trees*, in WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases, New York, NY, USA, 2004, ACM Press, pp. 25–30.
[6] V. GAEDE AND O. GUNTHER, *Multidimensional access methods*, ACM Comput. Surv., 30 (1998), pp. 170–231.
[7] P. GANESAN, B. YANG, AND H. GARCIA-MOLINA, *One torus to rule them all: multi-dimensional queries in p2p systems*, in WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases, New York, NY, USA, 2004, ACM Press, pp. 19–24.
[8] A. GUTTMAN, *R-trees: a dynamic index structure for spatial searching*, in SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data, New York, NY, USA, 1984, ACM Press, pp. 47–57.

[9] H. V. Jagadish, B. C. Ooi, and Q. H. Vu, *Baton: a balanced tree structure for peer-to-peer networks*, in VLDB '05: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, 2005, pp. 661–672.
[10] H. T. N. Jordi Pujol Ahullo, Ruben Mondejar Andreu, *Planetsim.* `http://planet.urv.es/planetsim/`
[11] A. Mondal, Y. Lifu, and M. Kitsuregawa, *P2PR-Tree: An R-Tree-Based Spatial Index for Peer-to-Peer Environments*, vol. 3268, Jan. 2004.
[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, *A scalable content-addressable network*, in SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA, 2001, ACM Press, pp. 161–172.
[13] A. Rowstron and P. Druschel, *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*, vol. 2218, Jan. 2001.
[14] C. Schmidt and M. Parashar, *Enabling flexible queries with guarantees in p2p systems*, Internet Computing, IEEE, 8 (2004), pp. 19–26.
[15] Y. Shu, B. C. Ooi, K.-L. Tan, and A. Zhou, *Supporting multi-dimensional range queries in peer-to-peer systems*, in Peer-to-Peer Computing, 2005. P2P 2005. Fifth IEEE International Conference on, 2005, pp. 173–180.
[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, in SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA, 2001, ACM Press, pp. 149–160.
[17] E. Tanin, A. Harwood, and H. Samet, *A distributed quadtree index for peer-to-peer settings*, in Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, 2005, pp. 254–255.

Edited by: Dana Petcu
Received: May 29, 2007
Accepted: June 19, 2007