Usage of Tree-based Indexing Scheme in Structured P2P System

Yi Yi Mar
University of Computer Studies, Yangon
yiyimar.yym@googlemail.com

Aung Htein Maw
University of Computer Studies, Yangon

Khine Moe Nwe
University of Computer Studies, Yangon

Abstract

Query processing is an essential role in large scale distributed network application environment including database indexing, distributed computing, location aware services and network monitoring system. In order to support complex queries including multi-dimensional and/or range queries, the efficiency of indexing scheme is important to be considered. This paper proposes the usage of tree-based multi-dimensional indexing scheme that is built over structured P2P overlay network. There are two phases in this indexing scheme (1) data locating phase using a balanced kd-tree (2) building of indexing mechanism which is based on the location of data on peers. This proposed indexing mechanism can support complex query processing over structured P2P overlay network and keeps load balancing among peers. In this paper, the performance of tree-based indexing mechanism is evaluated by using many simulated results.

Key words: Indexing over structured P2P system, DHT-based P2P system, Complex query processing over DHT

1. Introduction

Peer-to-peer architecture is widely used in many distributed applications because of its ability in good security and scalability [7]. Sharing content files containing audio, video, data or anything in a digital format is very common, and real-time data such as telephony traffic is also passed using P2P technology. In such applications, while complex query processing plays an important role for locating data by any peer in the systems efficiently and quickly [14], one challenge is organizing peers into a cooperative, global index. Another major challenge is load balancing of resource sharing among network nodes. There are many researchers proposed various techniques to enhance query processing in both unstructured and structured P2P systems.

Earlier peer-to-peer proposals were unstructured overlays like Napster [3] and Gnutella [2]. Unstructured P2P system can handle more complex queries such as multi-dimensional query, range query, keyword-based query. Napster is a centralized server providing a legal music service. A central server maintains the keywords as artist name, album title or song and information about all nodes and objects in the system. Even Napster has the simple routing mechanism with central server; it is poor in scalability and can also have a single point of failure. To overcome single point of failure, Gnutella is proposed as a fully decentralized protocol for file sharing. It uses flooding-based technique for communication and, as a consequence, they produce high message overhead as each node has to forward every message to every neighbor.

Structured P2P overlay network systems such as CAN [11], Chord [15], Pastry [12] and Tapestry [20] use distributed hash table (DHT) to provide exact match query with good performance quality. The DHT index allows the P2P system to sustain insert throughput and large data volumes, while ensuring fault-tolerance, and high availability. Most of the DHTs are one-dimensional indexing mechanism to locate data present in the system by implementing a Distributed Hashing Table. DHT is actually a data structure for storing of pairs (key, data) in a

distributed manner, which allows fast locating data when a key is given. For example, DHT based P2P systems can use the exact-match query interface with the filename as the keyword to publish and lookup files.

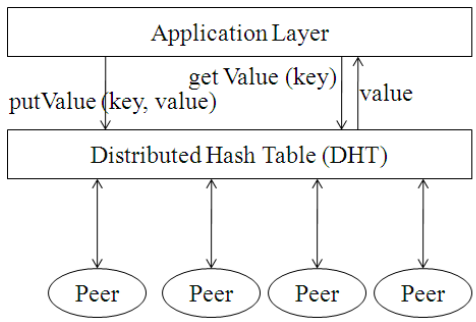


Figure 1.Structured P2P overlay network

Figure 1 shows the structured P2P overlay network. For an efficient information discovery service in distributed computing systems, the processing of complex query (multi-dimensional and/or range query) may be a challenge for DHTs. To address this issue, an indexing scheme is required to be built over underlying DHTs. In recent approaches, complex queries are provided by a single index which is created with the replication and combination of all attributes, and multiple indexes which combine the results of each attribute's index. One important fact is that any indexing scheme over DHT may also need to keep the load balancing among peers, i.e. one major issue in large amount of resource sharing in P2P system. This issue can be handled by using a balanced data structure over DHT [8].

This paper focuses on the usage of balanced kd-tree to build the proposed tree-based multi-dimensional indexing scheme over Chord. Optimal splitting threshold value ($\mathbf{T_{SP}}$) can make the kd-tree balanced. To keep kd-tree balance, in this paper, the difference $\mathbf{T_{SP}}$ values are tested with many simulated results.

The rest of the paper is organized as follows. In Section 2, the existing indexing approaches over DHT-based P2P systems are discussed as a related work. In Section 3, the architecture of the proposed indexing system is discussed and the required steps to build the indexing mechanism are described. The steps in complex query processing by using the proposed indexing scheme is discussed in Section 4. In Section 5, the required parameters for simulation setup are described. To show how kd-tree can affect the indexing scheme, the performance of kd-tree is also evaluated based on the various metrics in Section 5. This paper summarizes about the proposed indexing scheme in Section 6.

2. Related Work

In recent years, there are many researches for supporting query processing over DHT-based P2P environment. These indexing schemes use various data structures such as tree, graph, and grid.

Prefix Hash Tree, PHT [10] is the first indexing scheme over DHT that enables more sophisticated query. To process range query, PHT proposed two algorithms. The first algorithm resulted in high latency because all leaves are sequentially traversed until the query is completely solved. In second algorithm, it is parallelized and recursively forward the query until the leaf nodes overlapping the query. When the requested range is small, it may lead over loading.

To solve the overhead in PHT, DST [21] fill the internal nodes with data to violate traversing down to leaf node. So it stores keys not only in leaf nodes but also in internal nodes. To process a range query, it is decomposed into a union of minimum node intervals of segment tree. Then the query is solved by the union of keys returned from the corresponding DST nodes. However, it may leads maintenance overhead because keys are replicated over internal nodes and leaves. Distributed arbitrary segment tree,

DAST [4] is built for range query processing. It constructs an arbitrary segment tree and encapsulate the (key, data) pairs with segmentIds. When processing range query, it divides the requested query into the segments as in AST (arbitrary segment tree). And then it retrieves the data related with segmentIds. DAST can reduce the number of DHT retrievals. But AOR (accuracy of range) can drop because the union of segmentIds can also contain the irrelevant segmentIds.

Distributed Hilbert R-trees (DHR-trees) [18]

provides range query processing structure for P2P systems. It can achieve efficient query processing. But it cannot handle multidimensional query with one index, whereas it reduce the m-dimensions to one-dimension.

There is an indexing middleware that was proposed for DHT-based P2P systems. This system is performed over Pastry [12]. Unlike other systems, this system requires the object to register and each peer needs to keep meta-data of <attribute, value> pairs. Meta-data are fragmented and duplicated on peers. This system uses the meta-data to search the location (peerID) of requested data. This system can have an overhead of updating because meta-data of value are replicated over the peers.

In [17], m-LIGHT also used kd-tree to build efficient indexing scheme over underlying DHT. It proposes a new data aware splitting strategy to distribute data on kd-tree. And then it also proposed a new mechanism to map data from kd-tree to peer nodes. It is high efficient in query processing but still has the drawback in bandwidth and latency consuming.

In this paper, a tree-based indexing scheme with a balanced kd-tree over underlying DHT is proposed. This indexing scheme can support complex query and also can keep load balancing among peers.

3. System Architecture for Proposed Indexing Scheme

In this proposed tree-based indexing scheme, there are two phases.

[1] data locating phase

[2] building of indexing mechanism phase

In this paper, the real bibliographic dataset, DBLP [1] is used. DBLP data set is composed of bibliography data. It consists of published records for each author represented with multi-dimensional attributes, such as author name, year, title, book title, URL and conference. In these attributes such as author name, title, book title and URL are converted to floating points in the range of [0, 1] for data partitioning over kd-tree. To evaluate the performance of the proposed indexing scheme, DBLP dataset with three data sizes – 200 000, 500 000, and 700 000 is used, where data size is the total number of records in each system.

3.1. Data Locating to Underlying Network

Before distributing data among peers, balanced kd-tree is used to partition the large data set with the purpose of supporting data locality. Then the partitioned data are distributed among with optimal load balance. Kd-tree is a data structure and very useful in several applications, such as searches involving a multi-dimensional and range query [6, 9], geographic information system and computer graphic systems. Data locating phase, shown in figure 2, is an important role in this tree-based indexing scheme. The two steps are,

[1] Data partitioning using kd-tree

[2] Mapping data of kd-tree to overlay network

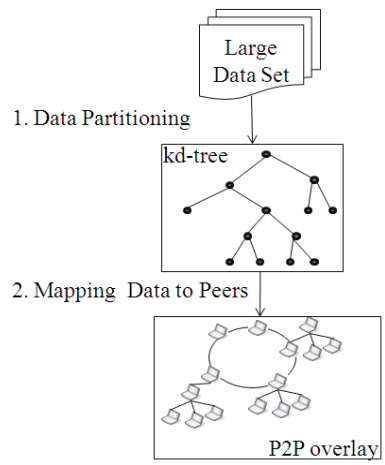


Figure 2.Data locating in P2P overlay network

3.1.1. Data Partitioning

In partitioning data, a balanced kd-tree is used. In figure 3, a balanced kd-tree is built according to two attributes or two dimensions, author name and year. Data are recursively partitioned into two sets along with different dimensions in an alternative fashion. The partitioning process continues until each node does not have the amount of data more than $\mathbf{T_{sp}}$. Half points of each dimension are generated while

building kd-tree. Data are only stored in leaves of kd-tree. In addition, half points of dimensions are also stored in leaves of kd-tree. In this indexing system, each peer needs to keep half points of leaves on kd-tree into local half point database (Hdb).

Labels or keys of data are defined while building kd-tree. Each node in kd-tree is labeled with "#" plus binary sequence "100..." The label of root node is "#" and the label of branch and leaf nodes are the concatenation of its own label (0/1) and the label of parent's node label. The label of left child is label of parent's plus "0" and right child has the label of parent's plus "1".

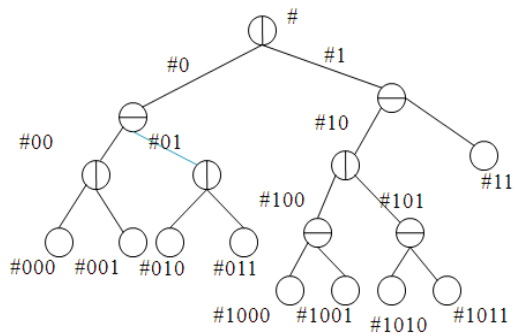


Figure 3.Kd-tree with labels

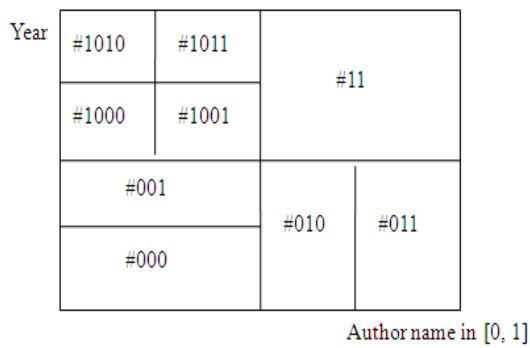


Figure 4.2-dimensional cell regions

In figure 4, leaves of kd-tree are represented in the form of rectangular cell regions. Each cell has a name with binary sequence. These names are labels of leave nodes of kd-tree. Each cell contains the related data (records) regarding with 2D partitioning. In this proposed indexing scheme, each leaf node's label is considered as a data key of data records in the same cell region. These data keys are used in mapping data to peers.

A major drawback in kd-tree may be highly imbalanced load on tree nodes [13]. An adaptive solution is to divide the data to two subgroups with equal amounts of data [19]. Optimal $\mathbf{T_{SP}}$ value can keep the load balance among kd-tree nodes. To get the optimal value, $\mathbf{T_{SP}}$ is tested by assigning the values between 100 and 1,000. In this paper, an optimal $\mathbf{T_{SP}}$ is defined by using the following two metrics

[1] number of empty nodes on kd-tree

[2] number of peers when data size is zero

After the kd-tree has been built, all data have data keys according the labels of leaves in kd-tree. To store data into DHT on each peer, data IDs are generated from hashing of the data keys by using a consistent hashing such as SHA-1 [5]. SHA-1 is a consistent hashing as which has good distributional properties. In this paper, data keys are values and data IDs are keys which are stored in DHT as pairs.

3.1.2. Mapping to Peer Nodes

This section discusses how to map the data keys of kd-tree to peers in an underlying overlay network. In this paper, the proposed indexing scheme is built over an underlying DHT overlay network, Chord [15]. Chord is a well known DHT. It uses keys to store and retrieve data. . In Chord overlay network, peer nodes' identifiers are organized in a ring topology. In order to distribute data among peers, data keys need to be mapped with the peer nodes in a balanced manner.

Random choice can provide balanced distribution [16]. So the set of keys and nodes are required to be randomly chosen. For this purpose, standard hash function can be used to distribute data keys and nodes IP hashed in randomness. Data IDs and peer IDs are computed by using SHA-1. In this paper, peer IDs are computed by hashing the IP address and data IDs by hashing of data keys. Then data IDs are distributed to the peer nodes whose IDs are closest (less than) or equal to the peer IDs.

Load balancing is an issue in any P2P system. Resource sharing among peers needs to be balanced. In this paper, we consider load on each

node is balanced while each peer has the load no more than T_{pl}, where T_{pl} is the maximum load on each node. If most of peers in the network hold T_{pl}, the P2P system will be balanced. T_{pl} can be computed according to (1), where T_r is the total amount of data in the system and N is the total number of peers in network.

$$T_{pl} = T_r / N \quad (1)$$

After mapping to peers, the overlay network can have the peers with zero data size where T_{pl} on peers is 0 because of consistent hashing. In this paper, T_{SP} is defined at the optimal value where a few number of peers with the zero data size.

3.2. Proposed Indexing Mechanism

The major objective of proposed indexing mechanism is to reduce the communication cost by generating locations of data in terms of data keys or labels before occurring DHT lookup operation. The proposed indexing mechanism can reduce the number of message forwarding or number of visited peers, and time consuming in query processing.

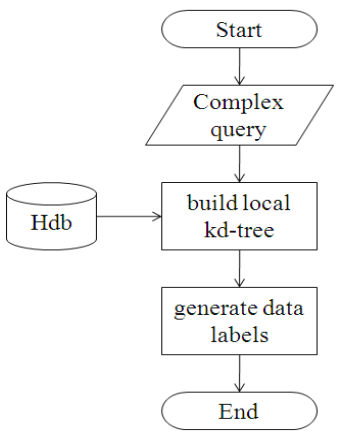


Figure 5.Process flow of the proposed indexing mechanism

Figure 5 shows the process flow of proposed indexing mechanism. This indexing mechanism starts when a complex query is requested. Then local kd-tree is built by using half points stored in Hdb. Then it generates the data labels or keys for the requested query.

4. Complex Query Processing in DHT-based Overlay Network

Figure 6 shows the step by step processing of a complex query over the proposed indexing mechanism.

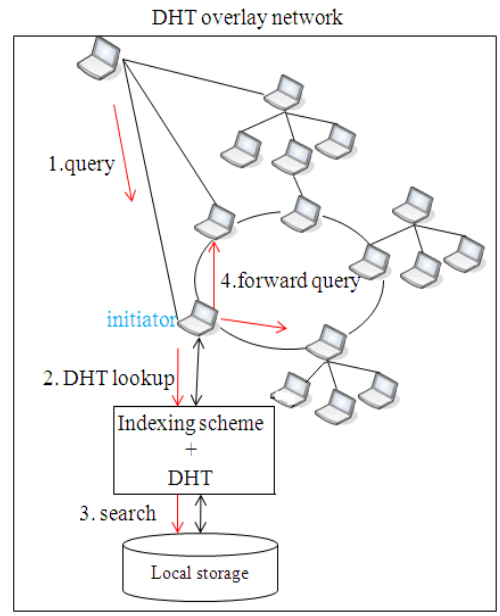


Figure 6.Indexing in DHT-based overlay network

When a query is sent to one of peers in the network, this peer works as an initiator. Firstly, the indexing scheme at the initiator checks in its own storage. If data are not found in local nodes, the indexing scheme generates the labels or keys of data. These labels can cover the requested query. Then this initiator forwards the labels of data to the peers where each peer's ID is closest the data labels. The forwarding occurs until the requested query can be retrieved.

5. Experimental Setup

The proposed indexing scheme shown in figure 6 is implemented with the use of java language as a simulated model. In this experiment, the required parameters are shown in table 1.

Table 1.Parameters for simulation

Parameters for overlay network simulation	
Number of peers in overlay network	1000
sizes of data source (1) DBLP-200 000 (2) DBLP- 500 000 (3) DBLP- 700 000	 (1) 200 000 records (2) 500 000 records (3) 700 000 records
Range of T_{sp} values	100 to 1000
dimension	multi dimensions

To keep kd-tree balanced, T_{SP} is defined via simulation results. This simulated model is tested with the various parameters as shown in table 1. In this paper, experimental simulations can demonstrate the effectiveness of the proposed approach. T_{sp} is considered based on three factors,

[1] kd-tree is balanced or imbalanced,

[2] the effects of kd-tree which can affect the resource sharing among peers with load balance or imbalance and

[3] how kd-tree can affect the indexing scheme while generation of labels (data localities) of requested query.

5.1 Performance Evaluation of Kd-tree

In this section, simulated experiments for kd-tree are shown. Kd-tree is used in various forms such as data structure only, an index for range searching, and so on. Therefore T_{SP} value can vary based on the applications. In this paper, kd-tree is used for data partitioning and supporting half points which are used in the proposed indexing mechanism. In this section, T_{SP} value is defined using three metrics to get a balanced kd-tree.

Firstly, T_{sp} is considered what value is optimal value to keep kd-tree balanced. After partitioning data on kd-tree, it becomes either balanced or imbalanced. Empty nodes are generated while the kd-tree is built. If there are large amount of empty nodes on kd-tree, it will be imbalanced.

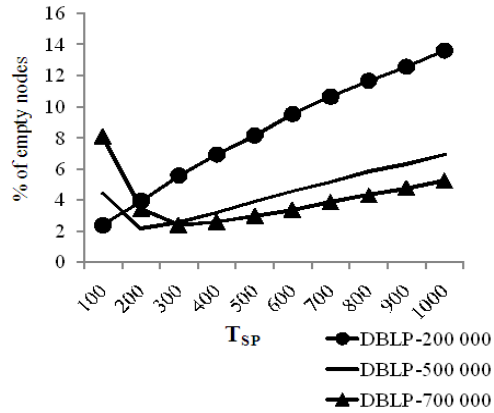


Figure 7.Percentage of empty nodes on kd-tree

As shown in figure 7, the percentage of empty nodes is the least at the value of T_{sp} with 200 while the data sizes are 500 000 and 700 000. While in data size 200 000, the least value occurred at T_{sp} with 100. According to these results, the value of T_{sp} that can keep kd-tree balanced is 200.

Secondly, T_{SP} is considered how it can affect the load balancing among peers. Load balancing is an important fact in P2P system. Therefore the indexing scheme over underlying DHT-based P2P system should keep load balancing. While mapping data among peers, some of peer nodes can have no data because of consistent hashing. The higher number of peers with zero data size can imbalance P2P network.

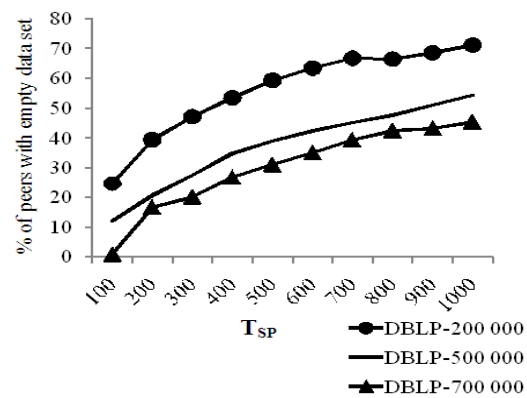


Figure 8.Percentage of peers with empty data set

Figure 8 shows that the greater the value of T_{sp}, the higher the percentage of empty peer nodes. For this case, the value of T_{sp} can only be the lowest value, 100.

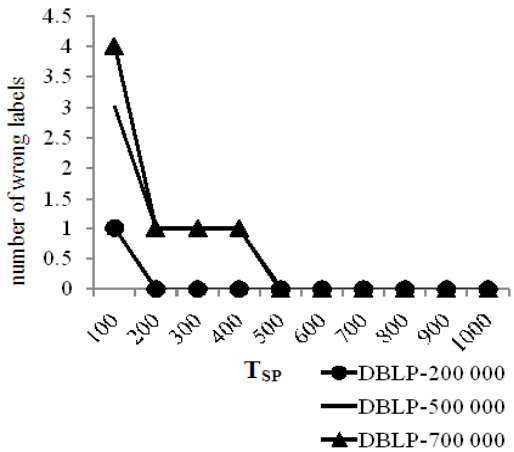


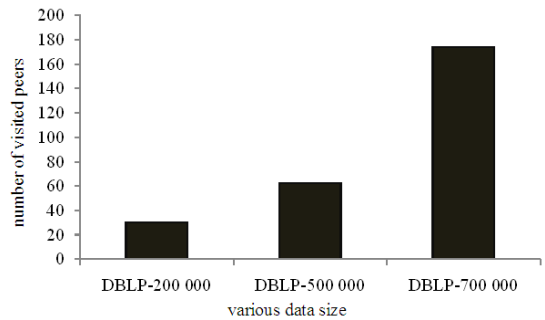
Figure 9.Number of wrong labels

Figure 9 shows the number of wrong labels generated by the indexing scheme. At data sizes 500 000 and 700 000, there are no wrong labels produced where T_{sp} is greater than equal 500. At T_{sp} 20000 and 400, the number of wrong label is one, and while at 100 with wrong labels more than one. At data size 200,000, there no wrong labels from 200 to 1,000, whereas one wrong label at 100. In our indexing system, query processing is depends on the labels or locality of data. Therefore the number of wrong labels can increase the number of DHT operations on irrelevant peer nodes. The indexing scheme can perform more efficiently if the number of wrong labels can be reduced.

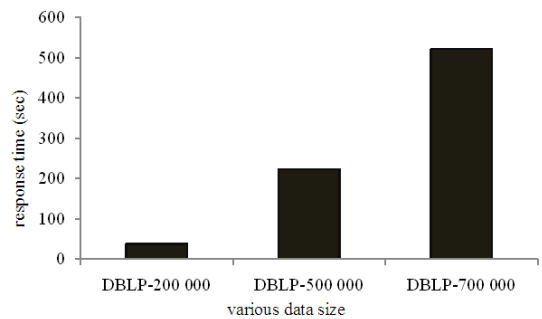
According the above figures (7), (8) and (9), T_{sp} value 200 is defined an optimal value. T_{sp} with can keep kd-tree balanced, load balancing among peers and can make our indexing scheme generate mostly true labels or data keys.

5.2 Communication Cost of Tree-based Indexing Scheme

Figure 10 shows the communication cost in terms of number of peers visited or number of message forwarding, and query response time or time consuming for a 4-dimensional range query. A 4-dimensional query is tested in this simulation model. This query is to retrieve a paper, i.e., "Generating a Device Driver with a Formal Specification Language" in the book of "Applied Informatics" by the author "Tetsuro Katayama" in the year "between 1990 and 2000"." This query is a 4-dimensional range query which uses four attributes such as author name, paper title, book title and year in the range of 1990 to 2000.



(a) Number of visited peers



(b) Query response time

Figure 10.Communication cost for a query for a 4-dimensional query

Figure 10(a) shows the number of peers visited for the above 4-dimensional query, where y-axis is the number of visited peers and x-axis is the various data sizes. Figure 10(b) shows the query response time, where x-axis is the various data sizes and y-axis is the response time perceived for this query. According to figure 10, the proposed indexing scheme is reliable because it can handle various amounts of data.

6. Conclusion

In this paper, the performance of tree based indexing mechanism is considered by building a simulated model. Based upon our simulation results, 200 is the optimal value for $\mathbf{T_{sp}}$ for building a balanced kd-tree. In this proposed system, there is no need to modify the underlying Chord DHT overlay network. In this paper, the indexing scheme can handle multi-dimensions (multiple attributes) by a single DHT-based P2P system. The evaluation results show that the proposed indexing scheme over structured P2P overlay network can also handle a large amount of data.

References

[1] DBLP. http://dblp.uni-trier.de/xml

[2] Gnutella. http://rfc-gnutella.sourceforge.net/

[3] Napster. http://www.napster.co.uk

[4] X. Chen and S. A. Jarvis, "Distributed Arbitrary Segment Trees: Providing Efficient Range Query Support over Public DHT Services", *The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07)*, 2007.

[5] FIPS. PUBS 180-2 Secure Hash Standard U.S. Department of Commerce/NIST, August 1, 2002.

[6] H. M. Kakde, "Range Searching using Kd Tree", 2005

[7] D. Liu and W. Xie, "Spatial Query on GroupP2P Networks", *In Proceeding of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing, Volume 02*, 2009.

[8] L. Lymberopoulos, S. Papayassiliou and V. Maglaris, " A Novel Load Balancing Mechanism for P2P Networking", *GridNets*, Lyon, France, Octorber 17-19, 2007.

[9] A. W. Moore, "An Introductory Tutorial on Kd-Trees", PhD. Thesis, University of Cambridge, 1991.

[10] S. Ramabladran, S. Ratnasamy, J. M. Hellerstein, and S. Shenker, "Prefix Hash Tree : An Indexing Data Structure over Distributed Hash Tables", *PODC*, 2004

[11] S. Ranasamy, P. Francis, M. Handley, R. Karp and S. Shenker, " A Scalable Content-Addessable Network", *SIGCOMM'01*, San Diego, California, USA, August 27-31, 2001.

[12] A. Rowstorn and P. Druschel, "Pastry: A Scalable, decentralized object location and routing for large scale peer-to-peer systems", *In Proceeding of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany, November 2001.

[13] S. Sarmady, "A peer-to-peer Dictionary Using Chord DHT", *Report*, School of Computer Science, University of Sains Malaysia, 2007

[14] S. Saroiu, K. P. Gummadi and S. D. Gribble, "Measuring and analyzing the characteristics of Napster and Gnutella hosts", *Multimedia Systems*, Volume 9 Issue 2, August 2003

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable Peer-to-Peer Lookup Service for Internet Applications", *In Proceedings of ACM SIGCOMM'01*, San Diego, September 2001.

[16] I. Stoica, R. Morris, D. L. Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications.

[17] Y. Tang, J. Xu, S. Zhou and W. Lee, "m-LIGHT: Indexing Multi-Dimensional Data over DHTs", *29th IEEE International Conference on Distributed Computing Systems*, 2009.

[18] X. Wei and K. Sezaki, "DHR Trees- A Distributed Multidimensional Indexing Structure for P2P systems", *Scalable Computing: Practice and Experience, Volume 8*, November 3, 2007, pp-291.

[19] M. Wu, "On R-tree Index Structures and Nearest Neighbor Queries", *Master Thesis*, University of Houston, December, 2006

[20] B. Y. Zhao, L. Huang, J. Strilling, S. C. Rhea, A. D. Joseph and J. D> Kubiatowicz, "Tapestry: A Resilient Global-scale over for the Service Deployment", *IEEE Journal On Selected Areas in Communications, Vol. 22, No. 1*, January 2004.

[21] C. Zheng, G. Shen, S. Li and S. Shenker, "Distributed Segment Tree: Support of range query and cover query over DHT", *In the 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2006.