A Peer-to-Peer Dictionary Using Chord DHT

Siamak Sarmady
School of Computer Sciences,
Universiti Sains Malaysia,
11800 Penang, Malaysia
{sarmady@cs.usm.my}

1 Introduction

Early P2P content distribution networks were relying on a central index of the available resources on the network. Due to problems happened to such services; completely distributed approaches are now being adopted. The new approach uses the concept of overlay networks to limit the amount of queries being spread over the network. Each host in this method chooses a few of the nodes on the network as its close neighbors. Transferring queries is then limited to a few of the neighboring nodes. Routing of the queries is being performed in a more efficient way to be able to provide both the scalability and guarantee of finding the resource.

1.1 Distributed Hash Tables

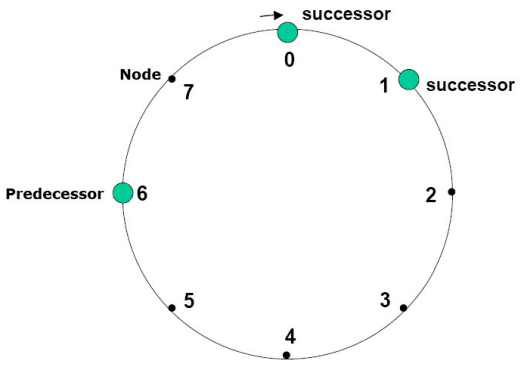
Distributed Hash Table (DHT) is able to fulfill two of our main requirements. DHT is a distributed data structure which holds key and value pairs in a completely distributed manner. It also puts each key-value pair on a single or limited nodes only (replication can be done to a few nodes to increase reliability). Each addition will only change data on a few nodes and the distribution of data is almost fair. To be able to determine on which node a specific pair should be stored we need a mapping method. In addition, adding and removing a node does not cause all the keys to be remapped. The mapping method being used in DHT is being called consistent hashing. Consistent hashing divides the key space into partitions. This method normally uses the concept of distance to map a key to a specific node. Distance is a logical concept and should not necessarily be related to physical or network distance of nodes. In a P2P network a node which is physically in Germany could be nearer to a node in Canada than a node in the same country. The mapping function is being used when we need to insert a new key-value pair into the hash table and also when we want to find the key. This function uses the key itself to determine the node which will store a pair. Then, when the same key is being queried, the same mapping function can determine the place which the key is being stored and therefore make retrieval of the value faster.

Using DHT each node when receiving a query tests to see if it has the key or not. If the key is not stored on the same node, the P2P application will use the mapping function to determine which of its neighbors has the least distance to the key-value's storing location. Then it forwards the query just to a few of its neighbors which are nearer to the storage location. Neighbors continue the same process until a node finds the key and sends it back. Insert process happens in a same way. When a node receives an insert request, it uses the map function to see how near it is to the keys place. Then it tests to see if any of its neighbors is nearer to the place which key should be stored. If one of the neighbors is nearer to the place, then it passes the pair to that neighbor. The neighbor repeats the process and passes the pair to one of its neighbors, which is nearer to the place. This process continues until a node is not able to find a neighbor which is nearer to storing place than itself (according mapping function). At this moment that specific node understands that, it is the best place to store the key and stores the key in its local database. To avoid loss of stored keys some implementations try to replicate their database to only a few of the nearest neighbors from time to time. In this way, we yet have little storage need on each node (if we wanted to replicate the entire content of the nodes on every node it would become impractical) and also the key-value pairs stored on removed nodes are preserved [3][4].

In this report we explore DHT technology by developing and testing a distributed dictionary which will hold words and their meanings on several peer-to-peer nodes. This system does not have dependence on a specific node.

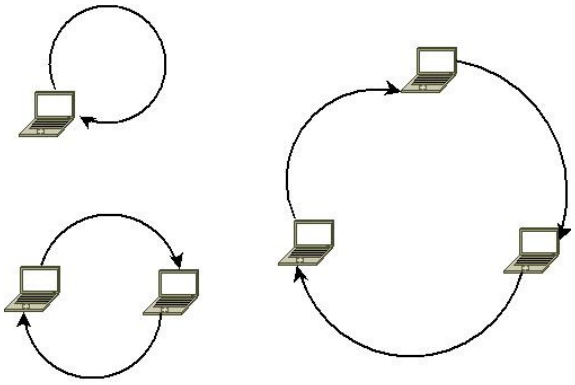
1.2 Chord Algorithm

Chord DHT algorithm has similarities uses "SHA-1" hash function as a base for consistent hashing. Nodes in Chord are placed on a Ring. Both node IDs and keys (hash from key-value pairs) are placed on the same ring. The hash function produces an m-bit identifier for both nodes and keys for this purpose. Each node has a successor and predecessor (Picture 1). Insertion of a new node between two older nodes involves the update of successor of one of those node and predecessor of the other. A key-value pair is assigned to the first node whose identifier is equal or follows the identifier of the key.



Picture 1: Chord Ring

If a node goes down, after a short period successor and predecessor nodes will discover this, using their Ping function and will update their successor and predecessor pointers. Responsibility of the key-space of the failed node will be transferred to the other nodes. Because each node in chord transfers queries to only one successor node, the flooding problem of the older resource discovery methods does not happen. [2]



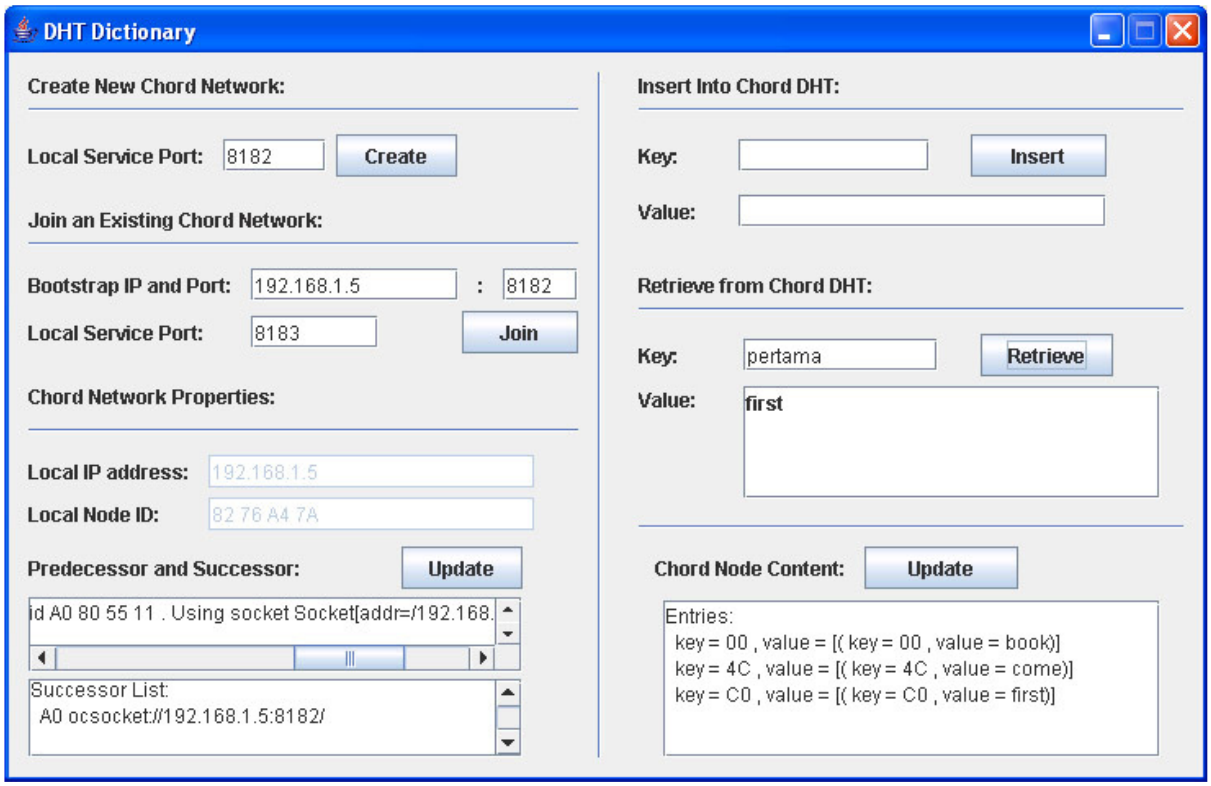
Picture 2: Smaller Chord Rings

Chord algorithm is able to handle a chord ring with even a single node (picture 2). In this situation all the entries are stored on that node until more nodes join the network.

2 DHT Dictionary Implementation

As mentioned earlier the core of our application is based on "OpenChord" Java library [1]. The user interface has been built using Java Swing library and therefore provides a GUI interface. Running the application needs a JRE version 1.5 or higher and a batch file "run.bat" has been provided for easier execution of the software. Picture 3 shows a snapshot of the software.

The software provides an option to create a new chord network or alternatively joining an existing network. After creating a network or joining an existing network, properties of this node on the chord ring network is displayed. This includes node ID, information about successor node and predecessor nodes and IP address of current node. Right side pane of the interface allows the user to insert a new word and its meaning to the chord network or to search for the meaning of a word in the distributed dictionary. A list of entries stored on each node (contents of the node) can be obtained using "Chord Node Content" section of the user interface.



Picture 3: User interface of the program

3 Experimenting with DHT

We use below list of words and their meaning in two scenarios to see how open-chord will distribute them between available nodes.

Satu	One
Dua	Two
Tiga	Three
Empat	Four
Lima	Five
Enam	Six
Tujuh	Seven
Lapan	Eight
Sembilan	Nine
Sepuluh	Ten

Table 1: Words and Meanings being used to Test DHT Dictionary

In scenario 1, we bring a single node up and insert all 10 words and their meaning into the distributed dictionary. Now we join 4 other nodes to the network one by one (we bootstrap each new node from one of previously up nodes). After joining each additional node we give the network a little time (5 seconds) to redistribute necessary contents if needed and then we list contents of each node. Table 2 shows that DHT with current configuration tries to maintain 3 copies of each entry so that if a node crashes, its content is not lost. (Table 2)

Online Nodes	IDs	Content of 1	Content of 2	Content of 3	Content of 4	Content of 5
1	A0	One Two Three Four Five Six Seven Eight Nine Ten	---DOWN---	---DOWN---	---DOWN---	---DOWN---
1,2	A0,82	One Two Three Four Five Six Seven Eight Nine Ten	One Two Three Four Five Six Seven Eight Nine Ten	---DOWN---	---DOWN---	---DOWN---
1,2,3	A0,82,AE	One Two Three Four Five Six Seven Eight Nine Ten	One Two Three Four Five Six Seven Eight Nine Ten	One Two Three Four Five Six Seven Eight Nine Ten	---DOWN---	---DOWN---
1,2,3,4	A0,82,AE,5A	One Two Three Four Five Six Seven Eight Nine Ten	One Two Three Four Five - Seven Eight Nine Ten	- - - - - Six - Eight - -	One Two Three Four Five Six Seven - Nine Ten	---DOWN---
1,2,3,4,5	A0,82,AE,5A,C8	- Two Three Four Five Six Seven Eight Nine Ten	One Two Three Four Five - Seven Eight Nine Ten	- - - - - Six - Eight - -	One Two Three Four Five - Seven - Nine Ten	One - - - - Six - - - -

Table 2: Ten entries being stored on five nodes. Nodes are added one by one.

In scenario 2, we bring all 5 nodes up initially and then enter 5 words. We investigate how these 5 entries are distributed between nodes. Then we bring down nodes one by one and investigate how content is maintained and redistributed on nodes. Again we see that DHT with current configuration tries to maintain 3 copies of each entry (if 3 or more nodes are available). As mentioned, when we remove nodes, content will be redistributed on nodes. Please pay attention that if multiple nodes are shutdown in a very short time, there is a risk that we lose some of the entries because DHT has not had enough time to replicate contents. We have therefore given sometime between shut down of each 2 nodes (10 seconds) in our test. (Table 3)

Online Nodes	IDs	Content of 1	Content of 2	Content of 3	Content of 4	Content of 5
1,2,3,4,5	A0,82,AE,5A,C8	- Two Three Four Five	One Two Three Four Five	- - - - -	One Two Three Four Five	One - - - -
1,3,4,5	A0,82,AE,5A,C8	One Two Three Four Five	---DOWN---	- Two Three Four Five	One Two Three Four Five	One - - - -
1,3,5	A0, AE, C8	One Two Three Four Five	---DOWN---	One Two Three Four Five	---DOWN---	One Two Three Four Five
3,5	AE, C8	---DOWN---	---DOWN---	One Two Three Four Five	---DOWN---	One Two Three Four Five

Table 3: Five entries stored on five nodes. Nodes are removed one by one.

4 Conclusion and Future Work

A sample application based on chord DHT was implemented and some experiments were performed on it. The work can be continued in many different directions. Comparison of the performance of different implementations of Chord algorithm and comparing the performance and features of chord algorithm to other algorithms can lead to a report which will help developers to choose the best solution for their needs. We are also interested in developing our own implementation of one of the available algorithms to get some hands on experience. Then we can possibly select a specific algorithm and introduce slightly different algorithms and compare the result of the changes applied. We can alternatively design a new algorithm and implement it.

5 Acknowledgment

I want to thank “Dr. Chan Huah Yang” for the “Advanced Distributed Systems” course I had with him and for the very enjoyable experience and valuable background it provided to me. I hope the knowledge will help me in future work and research in this field.

References

[1] "Open-Chord", http://sourceforge.net/projects /open-chord/, Accessed on August 2007.

[2] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", roceedings of ACM SIGCOMM 2001, San Deigo, CA, August 2001.

[3] H. Balakrishnan, M. FransKaashoek, D. Karger, R. Morris, I. Stoica, "Looking up data in P2P systems", Communications of ACM, Vol. 46, No. 2, Feb. 2003.

[4] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies", ACM Computing Surveys, Vol. 36, No. 4, pp. 335–371, December 2004.