iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 1 **Issue:** II **Month of publication:** September 2013

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

Advanced Chord Algorithm

Vikash Jaglan[1]

Dr. Sukhvir Singh[2]

Research Scholar, NCCE, ISRANA[1]

Head and Associate Professor, NCCE, ISRANA[2]

Abstract: *Network is a peer to peer (P2P) increasing popularity and now over a day. Systems on top of peer-peer overlay network abstraction of the machine or the physical network topology. Networks for structured and unstructured two types of networks are peer-to-peer name. The network peer to peer unstructured is not applied to cover the network structure. No network unstructured peer provide any organization or network connection optimized algorithms. The structural arrangement of such networks, according to specific rules or algorithms nodes provides specific performance and coverage topology. Therefore, the structure of the peer network coverage link to be set. They are typically distributed hash table (DHT) indexes, such as Chord system. Chord algorithm is relatively simple and successful alternative program to promote the diffusion of structured P2P network, but it must be better to further improve search efficiency. Therefore, this paper aims to improve the Chord structured P2P network discovery process. In order to improve search efficiency, the other to introduce this routing table in the counterclockwise direction, and the original routing table (in the clockwise direction search). Simulation results show that this method improves the work efficiency chord structured P2P network search process to reduce the number of hops.*

Keyword used: *P2P, DHT,*

1 Introduction

Becoming a structured peer network is becoming increasingly popular, because it has good performance and high scalability. Distributed hash table is used to generate less data traffic and prospecting program layout node. Chord protocol approach is popular and well-known DHT-based. Chord protocol is designed, simple protocol structured peer network architecture. Chord protocol is well distributed, scalability, stability, and load balancing. There are a lot of improvements and research Chord protocol

In this paper, we focus on structured peer networks and string algorithms Chord protocol performance and efficiency of the proposed changes to improve.

2 Chord Algorithms

Chord protocol is structured peer, solve problems, and effectively find the target node using the new approach. Cord protocol with provable performance, simplicity and provable correctness. Chord protocol is a key node mapping.

2.1 Properties of Chord

Properties of the chord peer-to peer networks are as follows:

a. **Decentralization:** Protocol string is not friendly, peer use of any central server or a super node. With others in the same system, the importance of each node. In this system, it has no single point of failure, the network is very strong [1]
b. **Availability:** Agreement in constant state of change for the network, it works fine: nodes can always find the way to the result, if there is no network fault, must surely be the key nodes and a large number of nodes network without adding chords. [1]
c. **Scalability:** Chord protocol can be used for very large systems, the cost of the logarithmic growth chords query protocol. [1]
d. **Load balance:** Since nodes use a key distribution cord same hash function. Therefore, the key is uniformly distributed in the node [1].
e. **Flexible naming:** The core structure of the cord without any restrictions, so its name is a great flexibility in the amount of data. [1]

Basic Chord Algorithm

Have some form of cord algorithm in this section is only a basic discussion of all form.

Chord algorithm [2] is a hash function, in network construction, resource discovery and resource information to join and processing nodes. The hash function can be changed at a fixed length output destination (node information, resource information). Hash function H (X) has the following properties:
H (x), and can generate a fixed-length output, where H (x) the operation of the data of any length. H (x) can be readily calculated for any specific x, where the process is reversed apparatus gained almost impossible to XH. So it is clear from the attributes hash functions can ensure uniqueness of data.

2.2.1Topological structure of Chord

Chord protocol service Stoica [3] proposed a distributed algorithm. Keywords are calculated taking into account the available data in the network resources and chords map contains peer. In topological string value of all peer> <IDK. IDK hash value is to find resources. Access to resources of the actual value of the storage location.

It is necessary to hash the string and each node of the network resources. The M-bit binary identifier idk as a result of the hash function. Hash node is the IP address of the node ID, and international domain names (IDN) represent the number of resource hash key same name, and is expressed in the IDK. IDnis range [0,2 M-1], internationalized domain names (IDN) arranged in a circle in the form of small macrocycles.

Figure 1.1 is an example of chord ring based on the basic chord algorithm with M=3.so maximum allowable node ID= (2^3=8).

The Construction process:

First, the node may need from a small ring to the line organization based IDNs (IDN). In Figure 1.1, the network nodes and keywords 0,1,3 1,2,4, IDK the IDN or equal to the first in a clockwise direction, the node is assigned to ID5.Keyword value. Successor nodes are called nodes IDK (successor (K)). As a follow-node (K) is equal to or less than the IDK IDK ring line, and set in a clockwise direction after the first node in the node. So, in the true capacity is a successor node (1) = 1, information, information resources on node 2 = 3 successor (2) Information Resource 4 and 5, the successor node (4); successor (5) = 0.

Figure 1.2 polyphonic ringtones sent to each node needs information to their successors to store information. In Figure 1.3, M = 6, so Chord network size = 26 = 64, but in fact only 10 nodes. These nodes are aware of the successor nodes. 8 is the information stored 14s node node node 21 node 14 and the like.The search process continues polyphonic ringtones for each node and its successor.

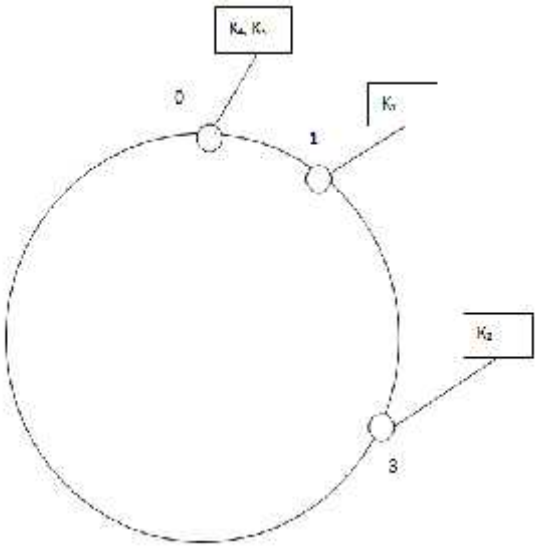


FIGURE1.1: CHORD RING M=3

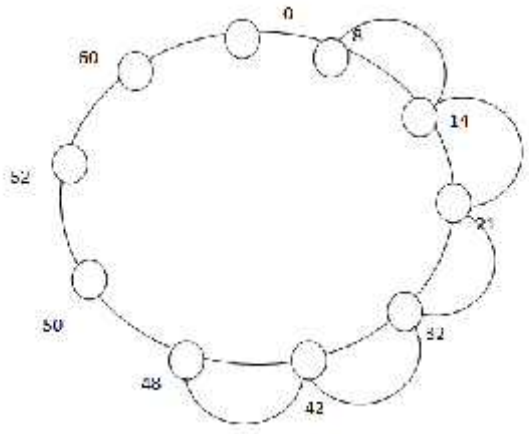


FIGURE1.2: NODE 8 SEARCHING IDK=45 IN BASIC CHORD

Flow chart for above algorithm is shown in figure 1.3.

According to flow chart in figure1.3:

Receive any node (n) in question (IDK) first checks, if IDK IDK = IDN or <IDn,successor(n)> node, if it is real or the result of the requested node or query message than the real successor (N), repeat this process for each node.

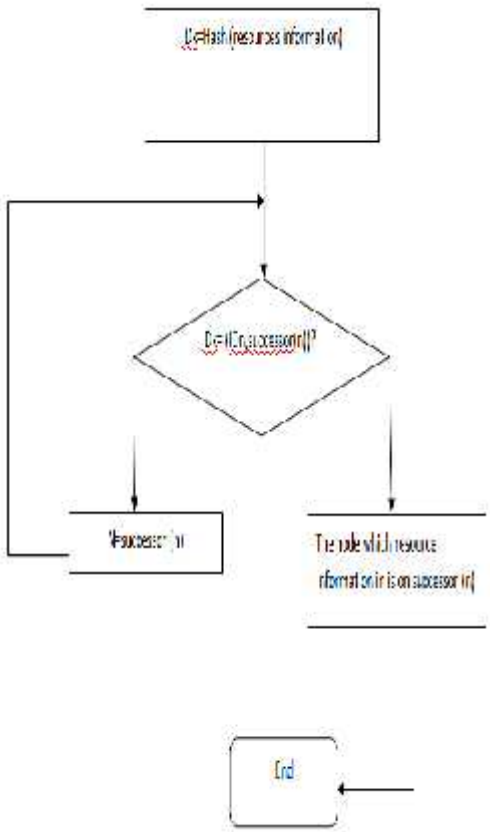


FIGURE1.3: LOOKUP PROCESS IN BASIC CHORD

The algorithm is a simple algorithm of the basic chords, and low efficiency because it requires multiple hops to reach the worst possible destination will node.tha n-1 times. Is an enhanced version of the chord algorithm, which is widely used now. The algorithm uses a fixed-length digital format, reducing the number of hops routing algorithm parameters.

2.2 Improvement of the chord algorithm

Because it is all ready to discuss your finger table maintenance agreements existing string and the search is completed clockwise order. Thus, the search time required node problem is the search. To search for the key that is located on the other side of the circular line, in the clockwise direction to scan network requirements. Therefore, the search process took a long time, it may reduce the effectiveness of the search process and the protocol string. Therefore, it is necessary to improve the string search algorithm to improve the efficiency and reduce the search time node in the ring. For our protocol strings need to consider the key elements to quickly find agreement and effectiveness.

In chord protocol each peer maintains the routing information so it possible to use this routing information to improve the chord searches efficiency.

Since time needed to search the key in any network can be as follows

$$\text{Time needed to find the keyword} = \text{average time per hop} \times \text{hops look for key word} \quad \text{____ (3.1)}$$

So reduction in the number of hops can reduce the search time.

Chord protocol restricting the use of paper in this project. String algorithms to solve the problems described above. In this project two directed search, will achieve two-way pointer table. Find a way to use the process to reduce the search time in the pointer table polyphonic ringtones. In other words, the concept is for each node in the routing table of the two existing tables: a first finger and the second finger table counter. By reverse pointer table each node can jump closer to the target peers.

There are two routing tables are implemented on each node

a) Figure table stores the successors and their key mapping for the node.

b) Anti finger table stores the predecessors of the node.

So finger table links the peer of the network in clockwise direction where as anti finger table links them in counter clockwise. The ways data are stored on nodes remain same. Hence, no change in the network's topological structure.

So in this improved chord algorithm there are three things maintain by each node

a) Finger table

b) Anti finger table

c) Successor list

Finger table and successor list is same as the original chord algorithm. This thesis have introduced new anti finger table.

There is no improvement if same half in ring have key .if key is in other half of the chord ring then has significant improvement in the lookup efficiency of the chord algorithm.

Following example explains the process

In figure 1.4 node 0 has predecessor node 6 and 7 and node 0 is in the finger table of the both table and hence both node 6 and 7 are in the anti finger table of the node 0.

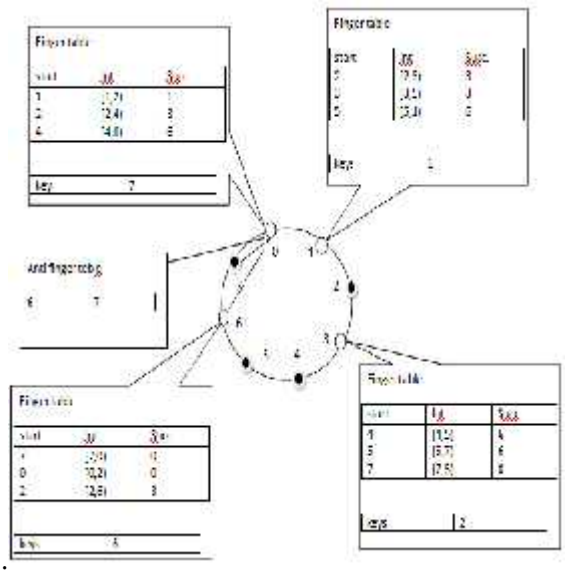


FIGURE 1.4: EXAMPLE

Lookup algorithm

In order not to change the original key and chord needs to change data mapping algorithm search algorithm. In can see the original algorithm, only the key chord successor. In the improved algorithm, we have a table to store the predecessor node fingerprint resistant, so you can not use the previous search algorithm to find a unique key. Use key fingerprint resistant sheet than its predecessor is the search for a successor. The original mapping key to maintain data.

Sometimes, it does not require all of the search process: the use of anti fingers. Use only fingerprint resistant sheet perspective, if a node in this table exceeds the value of the key. Node, which is used to find key data.

If the resulting fingerprint resistant data in the table, then the table with your fingers and search algorithms, according to the normal power cord. Search facilities clockwise and counterclockwise, the key can be found than the original algorithm.

2.4 Implementation

This section will discuss the algorithm to modify the original string OpenChord discussion. Openchord chord algorithm is a simulation platform open source. Finger table does not change, so do not require any modification of the finger table definition. It is known that m finger table entries. Table entry is similar finger against the finger table also m.

Algorithm used to calculate the number of hops the algorithm performance. Since implementation of the simulator used for small scale (<50 nodes), it is also based on the JVM (Java Virtual Machine). So polyphonic ringtones lookup speed is high, so we avoid the timer to calculate costs.

Pseudo code of proposed algorithm is:

List1.1: Pseudo code of the modified chord algorithm

```
Node findPredecessor(key,n){
    Node pred=n.getPredecessor();
        if (pred==null)
        return n;  //n is the current node
    else if (key.isInInterval(pred.ID, n.ID) //check if the key is between the pred and current node
        return n;
        else {
    Node n'=getClosestPrecedingNode(key) //if not, track the closest preceding node and lookup again
        return findPredecessor(key,n') // recursively find the predecessor of node n'
        }
}
Node findSuccessor(key,n){
        Node succ=n.getSuccessor();
if (succ==null)
        return n; //n is the current node
else if (key.isInInterval(n.ID, succ.ID) //check if the key is between the current node and successor's node
        return n;
        else {
Node n'=getClosestPrecedingNode(key) //if not, track the closest preceding node and lookup again
        return findSuccessor(key,n') // recursively find the predecessor of node n'
        }
}
Set<Serializable> retrieve_R(key){
        hops_R=0; //initialized the hops counter in anti-finger table direction
        while(!retrieved){
        Node responsibleNode_R=null;
        responsibleNode_R = findPredecessor(id);
        hops_R+=1; //while not retrieve the desired key, add the hop counter by 1
        try{
result_R = responsibleNode_R.retrieveEntries(id); // get the responsibleNode to fetch the entry
        retrieved = true; //if successfully get the value, set retrieved state to true
        }catch(Exception e){}
        continue;
        }
}
        if(result_R !=null) values1.add(entry.getValue()); // add the lookup result to the valueset
        final_hopsR=hops_R; //get the hop counter for the current lookup operation
        return values1;
}
Set<Serializable> retrieve(key){
        hops=0; //initialized the hops counter in finger table direction
        whild(!retrieved){
        Node responsibleNode=null;
        responsibleNode = findSuccessor(id);
        hops+=1; //while not retrieve the desired key, add the hop counter by 1
        try{
result = responsibleNode.retrieveEntries(id); // get the responsibleNode to fetch the entry
```

retrieved = true; //if successfully get the value, set retrieved state to true

}catch(Exception e){}

continue;

}

}

if(result !=null) values.add(entry.getValue()); // add the lookup result to the valueset

final_hops=hops; //get the hop counter for the current lookup operation

return values;

}

Obviously, the counter is set to jump from the pseudo-code your search results, search and search clockwise counterclockwise operation mode. Return value jump, eventually leading to the search process. Combined value will be helpful, to see or modify the chord algorithm performance comparison. Choose whether to continue automatically between clockwise or counterclockwise problem, we just get the next hop address using a lookup table and table antifinger methods fingers original chord algorithm.

List1.2: Pseudo code of lookup method of chord algorithm

```
// ask node n to find the successor of k
n.find_successor(k) {
    if (k∈(id(predecessor), id(n)])
        return n;
    else
        if (k∈(id(n), id(successor)])
            return successor;
        else {
            n'=closest_node(k);
            return n'.find_successor(k);
        }
}

// search the local routing table for the closest
// node to k, R denotes the routing table
n.closest_node(k) {
    x=R(1);
    for i=1 to |R|
        if (|R(i)-k|<|x-k|)
            x=R(i);
    return x;
}
```

Using the above method using the X1 and X2 on the query to the next node in a clockwise direction and the counterclockwise direction finding. After that calculate the distance between X1 and X2, we look forward questions less than the distance from the destination node node.
Like D1 = | X1-K | D2 = | X2-K |, if d1 is less than d2 question to the X2 X1 otherwise.
By using this method selectable forward direction or in the other direction can be automated.

3.1 Results

Open chords for the implementation and performance evaluation algorithms simulator modified chords. Will be used to improve the Chord algorithm and the original Chord algorithm to compare the performance of the two algorithms.

Test Results

The tests results will be show in term of two values .which are as follows:

a) Routing table size

Modified algorithm routing table has no entry in the program size fingerprint resistant finger table and the total number of entries. Because it is all ready know, not the finger table entry is less than or equal to M. In the experiment, the network is to simulate different node numbers and calculate two tables fingers and fingerprint resistant entries in the table, in hops clockwise and counterclockwise. The size of each peer routing table almost 2M.

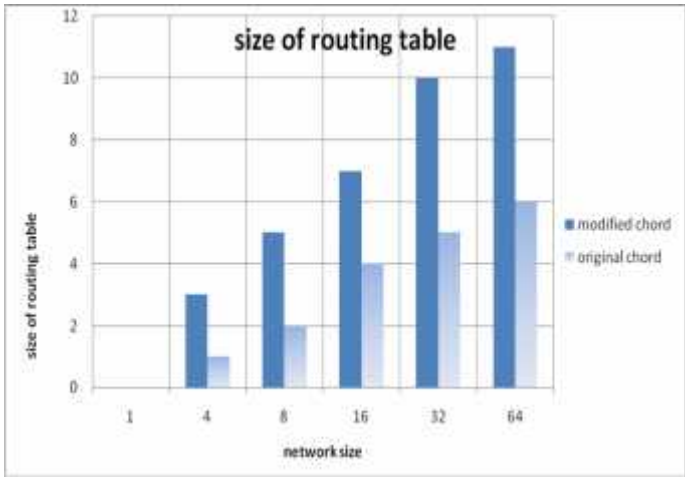


Figure1.5: comparison of routing table size for original and modified chord algorithm

b) Look up Hops

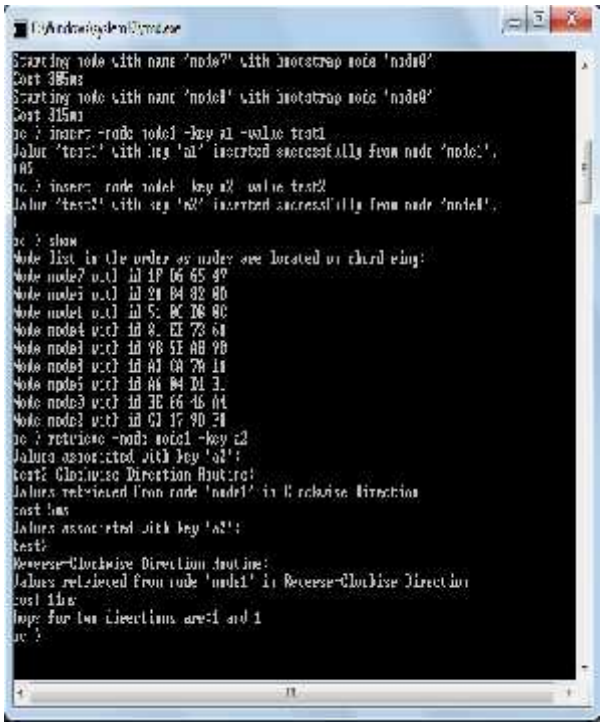


FIGURE1.6: SCREENSHOT #2

Search process performance algorithm to calculate the modified string comparison, we chose the original chord find and modify the look and recorded on the same node and search the same key. Inspection process, for different network size.

No of nodes	Entries in finger table	Size of finger table	Hop count
1	0	0	1
4	2	8	1
8	3	24	1
16	4	64	1
32	5	160	2
64	6	288	2

TABLE1.1: EVALUATION TABLE FOR CHORD ALGORITHM

In table 1.1 numbers of entries and size of the finger table are calculated and also calculated the hop count for different network size using original chord algorithm.

No of nodes	Entries in finger table	Size of finger table	Entries in anti finger table	Size of anti finger table	Exp. num. (clockwise direction)	Hop count(anti clockwise direction)	Total entries in routing table
1	0	0	0	0	1	1	0
4	2	8	2	8	2	1	4
8	3	24	3	24	3	1	6
16	4	64	4	64	4	1	8
32	5	160	5	160	5	1	10
64	6	288	6	288	6	1	12

TABLE1.2: EVALUATION TABLE FOR MODIFIED CHORD ALGORITHM

Table 1.2 shows result for modified Chord algorithm. It is clear from table3 that, use of anti finger table, make it possible to reduce hop count to search any key. To get the better result test this in an environment which is capable of generates nodes equal to the actual network.

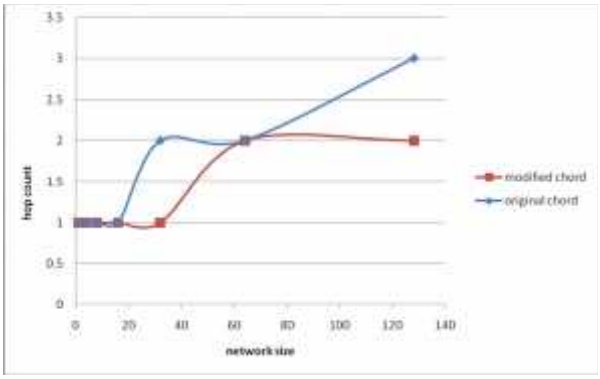


FIGURE1.7: COMPARING HOP COUNT FOR ORIGINAL AND MODIFIED CHORD

In the graph in Figure 1.7 shows the algorithm less hops Cord better original Chord algorithm as the network size is increased compared. Thus, the experiment showed that the modified algorithm Cord cord than the original search algorithm improved efficiency, better performance is.

4. Conclusion

The original chord key table search algorithms use a finger in one direction ie clockwise. It takes more time to find any hops to search for a particular key is on the other side of the ring line thesis proposal Chord routing table search algorithm in both clockwise and counterclockwise directions. For this purpose, he used a finger fingerprint resistant tables and tables. Appearance chord algorithm efficiency.

Open chords chord algorithm simulator for simulating changes. By simulating a modified conclusion Hops Chord algorithm, the need for specific search network to reduce the number of keys. In Chapter 5, it shows that as the network size increases, more hops to search for the key ring lying on the other side of the table by using only your fingers, but using table antifinger can search for a particular key No fewer hops. Therefore, the proposed algorithm Chord chords algorithm efficiency and reduce the search time.

5. References

[1] I. Stoica R, Morris D, Karger M Kaashoek F, Balakrishnan H, Chord: a scalable peer-to-peer lookup service for Internet applications, In Proc ACM SIGCOMM01, San Diego, CA, Aug, 2001.

[2] De-gan Zhang, Yu-xia Hu,Dong Wang and Yan-pin Liang, A New Algorithm of Service Discovery Based on DHT for Mobile Application, JOURNAL OF NETWORKS, VOL. 6, NO. 10, OCTOBER 2011.

[3] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma and Steven Lim, A Survey and Comparison of Peer-to-Peer Overlay Network Schemes, IEEE COMMUNICATIONS SURVEY AND TUTORIAL, MARCH 2004

[4] R. Denneman, P2P searching methods, research issues, solutions and their comparison, 11th Twente Student Conference on IT, Enschede, June 29th, 2009

[5] John Risson, Tim Moors,Survey of research towards robust peer-to-peer networks: Search methods,ScienceDirect, Computer Networks 50 (2006) 3485–3521

[6] Chen Gang, Wu Guoxin, Yang Wang. “G-Chord: an improved routing algorithm for Chord”. Journal of southeast university (Natural Science Edition), Vol.37, 9-12. Jan. 2007.

[7] Jiajing Li, Xu Dongyang. “An Optimized Chord Algorithm for Accelerating the Query of Hot Resources”. International Symposium on Computer Science and Computational Technology, Vol.2, 644-647. Dec. 2008

[8] Hong Feng, Li Ming-Lu. “SChord: Handling Churn in Chord”. Journal of Nanjing University (Natural Science Edition), Vol.41, 288-293. Oct, 2005.

[9] Sonia G, Habib Y. “Improving Chord Network Performance Using Geographic Coordinates”.

Proc of the 3rd International Conference of GeoSensor Networks. Berlin: Springer, 2009.

[10] Wang Biqing, He Peng. "L-Chord: Routing Model for Chord Based on Layer-Dividing". International Conference on Computational Intelligence and Security, Harbin, China, Dec. 2007: 262-265.

[11] Hongwei Chen, Zhiwei Ye. "BChord: Bi-directional routing DHT based on chord". 12th International Conference on Computer Supported Cooperative Work in Design, Xi'an China, April.2008: 410-415.

[12] Ashok Kumar Ojha, Krishna Kant. An Efficient Lookup Algorithm for Dynamic Peer-to-Peer Chord . IEEE2010

[13] Shunli Ding, Xiuhong Zhao .Analysis and improvement on Chord protocol for structured P2P . IEEE2011

[14] (Mo Hai, Shuhang Guo. A General Search Method Based on Social Communities in P2P Networks . IEEE2010

[15] "PeerSim P2P Simulator," 2005, accessed 01-May-2006. [Online]. Available: http://peersim.sourceforge.net/

[16] K. Shudo, "Overlay Weaver," 2006, accessed 01-May-2006. [Online]. Available: http://overlayweaver.sourceforge.net/

[17] http://open-chord.sourceforge.net/

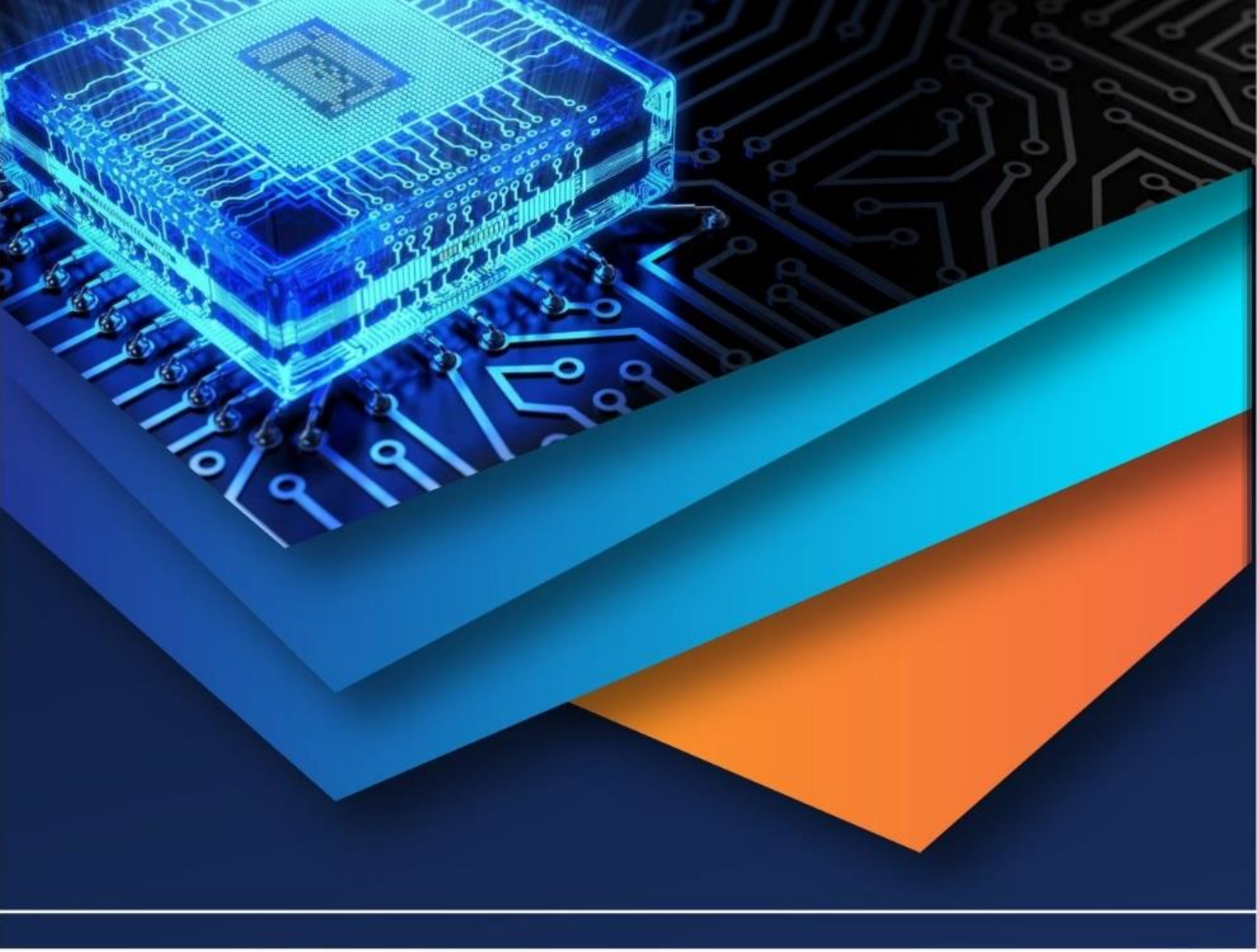
[18] http://en.wikipedia.org

[19] (2001) Kazaa media desktop. [Online]. Available: http://www.kazaa.com/

[20] Napster. [Online]. Available: http://www.napster.com/

[21] (2001) Gnutella development forum, the gnutella v0.6 protocol.[Online]. Available: http://groups.yahoo.com/group/the gdf/files/

[22] Gnutella website: http://www.gnutella.com



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY