International Journal of Computer Science Trends and Technology (IJCST) – Volume 7 Issue 2, Mar - Apr 2019

RESEARCH ARTICLE OPEN ACCESS

A Study to Improve Performance of Chord Algorithm in DHT-based Structured P2P Networks

Wael Hwiaje [1] Dr.Talal Alaateky [2]
Department of System and Computer Networks Engineering
University of Tishreen
Lattakia - Syria

ABSTRACT

Because of the importance of routing algorithms and the location of keys in DHT Structured P2P networks, including the Chord algorithm on which effective applications are built in different fields, and the importance of maintaining the stability of the network despite the continuous dynamic in which the joining and exit and failure of the peers . For That this study was to evaluate the performance of the original Chord algorithm and to try to improve the performance of the Chord algorithm after the algorithm code obtained by the OverSim simulation tool was obtained .

Modifications were made in the Code of the Chord algorithm associated with the search mechanism used in the nodes finger tables. These modifications, according to the experimental results, improved the access to the desired key by reducing the number of hops and thus reducing the time required to perform the Lookup process.

Keywords:— anti finger table, P2P, Structured , DHT, Lookups , RandomChurn , finger table, hopCountMax , finger table, NoChurn , Replication , predecessor , successor , OMNET++ , MaxPeer , OverSim

I. INTRODUCTION

The networks systems across history have evolved according to multiple stages to meet the needs of users and to increase the reliability of the network . Figure 1 illustrates these stages of the first phase , which witnessed the emergence of the primitive systems ARPANET , Usenet through the second phase in 1995 and the emergence of the client-server system and subsequent applications On this system from the Web , FTP , to the most important stage of the emergence of P2P networks in 1999 with their generations , Napster , Gnutella , Kazaa different differences between them until access to the most important generation of P2P networks is the generation of networks DHT-Based Structured P2P , which belongs to our studied Chord algorithm [10]

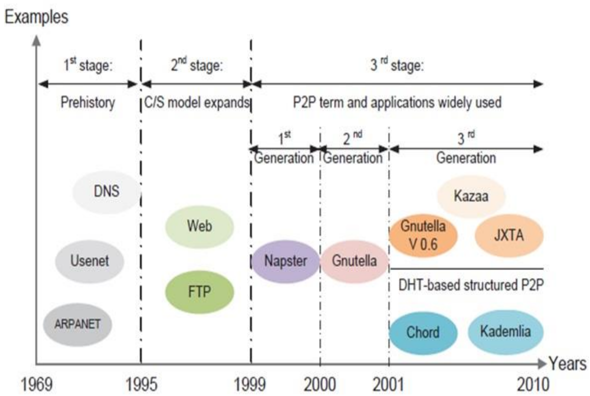

Figure 1 Timeline for evolution of network systems

Client-Server , P2P

II. STRUCTURE OF THE ALGORITHM

The Chord algorithm is based on a structure to regulate nodes and keys within the network to ensure that the keys are distributed on the nodes fairly evenly. It provides proper access to the desired key according to a specific mechanism and precise routing information. This algorithm is based on a circular structure where the nodes and keys that derive their positions Depending on the hash SHA-1.[1]

In order to verify the Chord algorithm, it is based on the parameter m, where the peers and keys distribution is achieved based on the following relationship (modulo 2 ^ m). Hash SHA-1 ensures that a unique identifier for each node and key in the system is as follows:[7]

ID(node) = SHA-1 (node IP)
ID(key) = SHA-1(key)

The keys are placed on nodes based on the rule: **The identifiers are arranged in circular form and modulo 2 ^ m** , The k key is placed on the first node whose identifier equals or follows the identifier of the key K, and that node n is called the successor node of the k key (the subsequent node) Successor (k) = n .

Thus, you can say that the k key will be placed in the next node clockwise from the node that has an identifier equal to the same k key.[15]

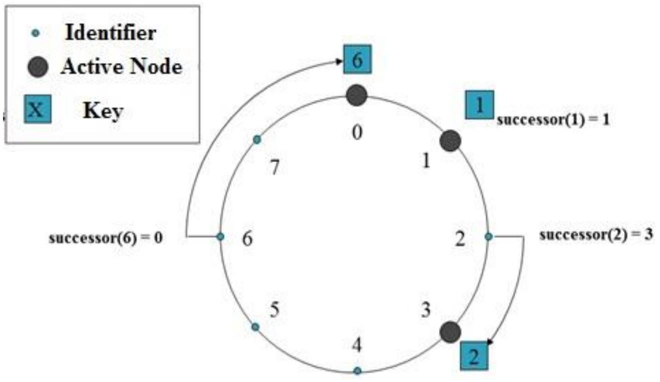

Figure 2 Example of Chord algorithm structure[14]

III. ROUTING IN THE CHORD ALGORITHM

Based on the above, each node must be aware of the following node in the network until each query is answered for a required key . However, this mechanism may require that all nodes in the network be sequentially accessed until the desired key is reached .[2]

To solve this problem, routing information has been added to each node, ensuring correct access to each key as long as this information is correct and updated constantly to keep up with any change in the network . [8]

Notation	Definition
Finger[k].start	$(n+2^{i-1})$ Mod 2^m 1 ≤K≤m
. Interval	[finger [k].start , finger [k+1].start]
.node Successor	First node ≥ n.finger [k].start
Predecessor	The previous node on the identifier circle

Based on the previous table, we have the finger tables shown in Figure 3 of the previous network , which includes the nodes (0, 1, 3) and keys (1, 2, 6) so that each node has a table of 3 lines , which includes a specific area of nodes and a node responsible for that domain, as well as the keys assigned to that node.[11]

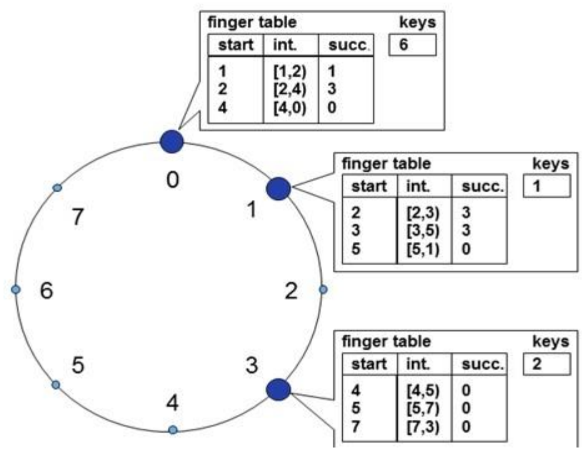

Figure 3 Chord algorithm structure with finger tables [11]

Based on the nodes finger tables , it is noted:

- Each node knows more about the neared nodes to it and followed in the order of the circle more than the nodes away from it .

- Depending on the table for node n , it does not have to contain the Successor of a given random k key (ie node n is directly aware of where the K key is located) , then node n searches for a node whose ID is closer to the k key , so it searches its node table for j node that have its identifier preceded by k and the query is then moved to look for Successor (k) . By repeating this process, n Knows where the key K is located.

- To illustrate this mechanism we will assume that node 3 wanted to look for Successor for key 1 , it will look within its finger table as shown in Figure 4 below . ID 1 is within interval = [7,3) , which is within the third line of the finger table as shown in step 1 of Figure 4, and Successor for this domain is node 0 , and ID 0 precedes 1, so node 3 asks node 0 to find Successor (1) (Step 2 of the figure 4). Depending on node 0 finger table , we will find that Successor(1) = 1 , therefore returning node 1 to node 3 as shown in step 3 of Figure 4 . [8]

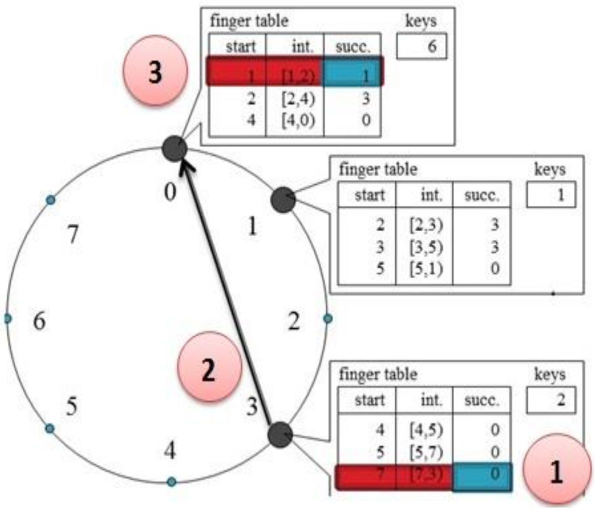

Figure 4 Search mechanism in the finger table of node 3

IV. MODIFIED CHORD ALGORITHM

The previous search process used in the Chord algorithm can be summed up in the programming structure described in Figure 5 . In my study of the code structure used to search within the Finger tables, the search for the successor of the key (id) within the node's table takes place from the last line In the table, as if assuming the worst case (to be far from the node), but the search result can be in any line, resulting in a delay in the search if it is in the first lines and search is from the last line.

Thus , to reduce the search time, it is assumed that it is better to search from the middle of the table and move up or

```
// ask node n to find id's successor
n.find_successor(id)
   n' = find_predecessor(id);
   return n'.successor;

// ask node n to find id's predecessor
n.find_predecessor(id)
   n' = n;
   while (id ∉ (n', n'.successor])
      n' = n'.closest_preceding_finger(id);
   return n';

// return closest finger preceding id
n.closest_preceding_finger(id)
   for i = m downto 1
      if (finger[i].node ∈ (n, id))
         return finger[i].node;
   return n;
```

down accordingly, in an attempt to reduce the search time and the number of hops.

Figure 5 The programming structure of the search mechanism in Finger tables (Original Chord) [8]

In this modification to the algorithm of the Chord algorithm, the search was carried out from the middle of the routing table for each node, in a sequential attempt to divide each routing table into two halves, performing the search in the best half of the search process, ignoring the other half and thus attempting to search for the desired key through the information To illustrate this difference, and to illustrate the difference between the mechanism used in the original and modified chord equations, here are examples Figure 5 and Figure 6 illustrating the programming structure of each algorithm .

```
// ask node to find id 's successor
n.find-successor (id)
ń= find-predecessor (id);
return ń.successor ;

// ask node to find id 's predecessor
n.find-predecessor (id);
ń=n;
while ( id ¢ (ń, ń.succesor])
ń= closest-preceding-finger (id);
return ń ;

// return closest finger preceding id
n.closest-preceding-finger(id)
{ i= m div 2 ;
  IF (finger[i].node ϵ (n , id) )
    { for  h=m downto i
        if (finger[h].node ϵ (n,id) )
          return  finger[h].node ;
    } else
  for i= i downto 1
     if (finger[i].node ϵ (n,id))
     return  finger[i].node ;
return   n ;
}
```

Figure 6 The programming structure of the search mechanism in Finger tables (Modified Chord)

V. OVERSIM SIMULATION TOOL

To perform the research, a simulation tool must be found to verify the original Chord algorithm , and provides The full code for the algorithm that the researchers did not publish that fulfills the Chord algorithm, so the chosen simulation tool should have the following advantages: [12]

- Provide high dynamic network in the cases of joining and exit nodes.
- Scalability in the number of peers , because the study assumes the construction of a large network somewhat .
- An open-source tool that allows for new protocols.
- The possibility of achieving the Replication of the keys in both the previous node and the next, in order to ensure access to the keys permanently, despite the failure at high rates.
- Ability to modify and improve the original chord algorithm.

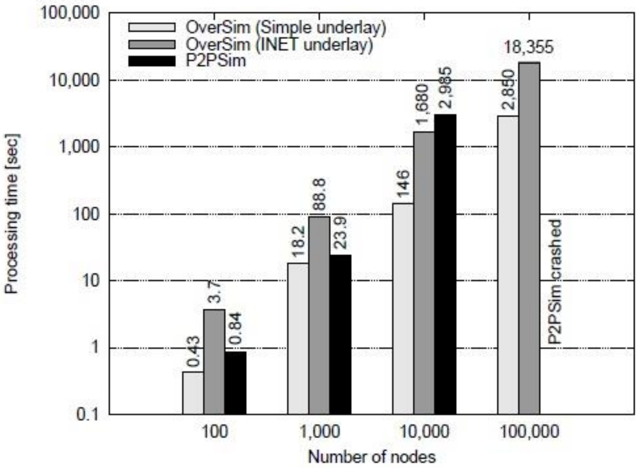

Figure 7 Performance evaluation of OverSim, P2PSim simulation tools in terms of processing time [13]

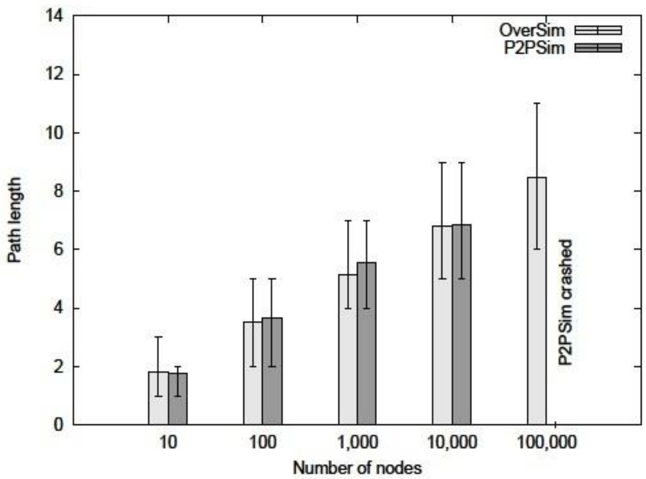

Figure 8 Performance evaluation of OverSim , P2PSim simulation tools in terms of path length [13]

Many simulation tools were developed (OverSim , P2PSim, PeerSim, PlanetSim) . OverSim was chosen as a simulation tool to complete the search after a long effort to find the complete algorithm code. The OverSim simulation tool, based on Omnet ++, achieves the chord algorithm, and the OverSim simulation tool is the best among them, which achieves less processing time and shorter access to the Key target. As shown in Figures 7, 8 .
The simulation tool has been activated and is able to handle the number of nodes specified for the study. Figure 9 shows the status of the network when reaching 2054 nodes within the system.

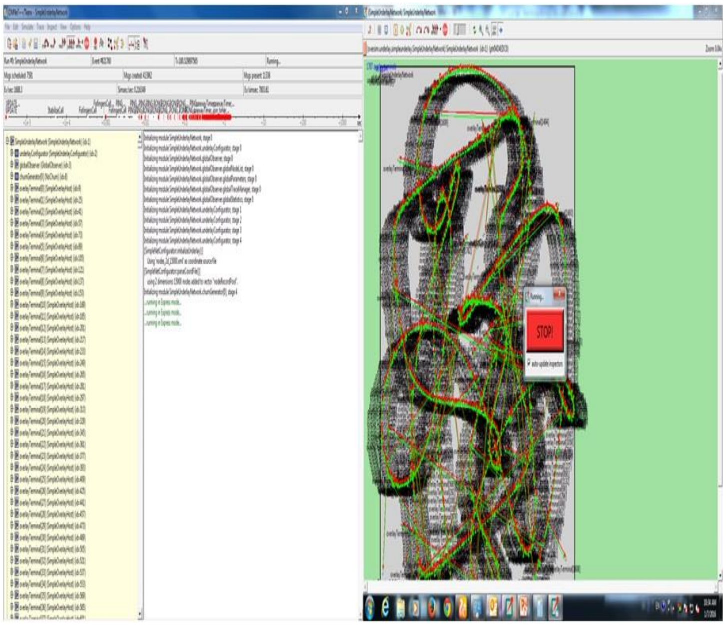

Figure 9 Chord network architecture according to the OverSim simulator when reaching 2054 node

VI. EXPERIMENTAL RESULTS

Experimental determinants: The network was studied in the case of increasing the number of nodes N = 250 to 2000, and given the number of keys K = 2 * N, Lookups = 20 (number of key extraction operations), and to give accurate results on the effect of this modification in improving the performance of the algorithm ؛The network is considered to be in a state of permanent stability where there is no sudden exit of the nodes , using (NoChurn node behavior) where the number of nodes in the network increases to reach targetOverlayTerminalNum (N) and then the network is in a steady state.

Considerated factor : **hopCountMax**

Name	Type	Description
hopCountMax	**Int**	**maximum number of overlay hops**

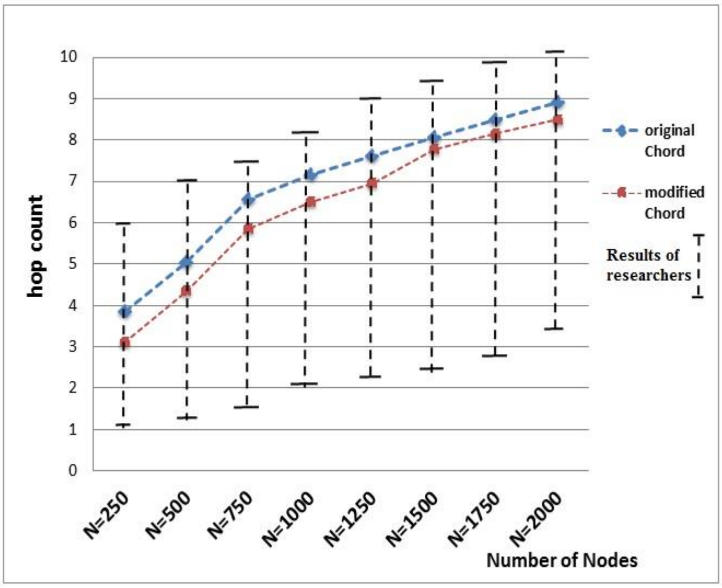

Figure 10 The average number of hops in the original and modified chord

VII. CONCLUSIONS

Note that the editing within the search mechanism in the nodes Finger tables that described in the procedure **[NodeVector*Chord :: closestPreceedingNode (const OverlayKey & key)]** , and shown in Figure 6 in this paper, contributed to the improvement of the number of hop by **19.48%** at number of peers is 250, and gradually decrease the rate of improvement until the rate of improvement of **4.944%** at the number of nodes examined maximum 2000 nodes.

Based on the above results and comparisons, we recommend that you adopt the new search mechanism within the modified Chord algorithm for its ability to achieve faster access to the desired key through a better hop rate and thus ensure the fastest time to perform lookups.

ACKNOWLEDGMENT

The research that led to this work was supported in the first part by the Tishreen University Foundation .
The authors wish to thank the University administration and the College of Informatics Engineering for their assistance in providing technical support for this work .

REFERENCES

[1] Vikash Jaglan , Dr. Sukhvir Singh , Advanced Chord Algoritthm , International Journal For Research IN Applied Sciense And Engineering Technplpgy

(IJRSSET) . Vol. 1, Issue II, September 2013, ISSN: 2321-9653

[2] Chunling Cheng , Yu Xu & Xiaolong Xu . “Advanced Chord Routing Algorithm Based on Redundant Information Replaced and Objective Resource Table “.Augest 2010 .

[3] Diana Chinces & Florian Mircea Boian , Optimizing the search operation in P2P networks. ARPN Journal of Systems and Software , Vol. 1 No. 1, APRIL 2011 , 24-27

[4] Sabu M. Thampi & Chandra Sekaran. K , SURVEY OF SEARCH AND REPLICATION SCHEMES IN UNSTRUCTURED P2P NETWORKS , Vol. 2, No. 1 , 2010 , 93-131

[5] Zheng, X & Oleshchuk, V. Improving Chord Lookup Protocol for P2PSIP-Based Communication Systems . New Trends in Information and Service Science, 2009. NISS '09. International Conference , June 30 2009-July 2 2009 , 1309 – 1314 .

[6] Biqing Wang & He, Peng , L-Chord: Routing Model for Chord Based on Layer- Dividing. Computational Intelligence and Security, 2007 International Conference , 2007 , 262 – 265

[7] Wang Ziyang & YYin Siqing . Chord algorithm based on two-way routing table. Computer Application and System Modeling (ICCASM), 2010 International Conference on . Vol. 7, 2010 , 168-171 .

[8] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H.Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, Proc of the 2003 ACM Annual Conference of the Special Interest Group on Data Communication (ACM SIGCOMM’01), 2003

[9] Song and Shang Gao, “ Improved Chord Routing Algorithm Based on the Finger Table “, Journal of Communication and Computer 10 (2013) 717-719

[10] LUA,E.K & CROWCROFT,J , PIAS,M & LIM,S . A Survey and Comparison Of Peer-To-Peer Overlay Network Schemes.IEEE COMMUNICATIONS SECOND QUARTER. VO. 7, NO. 2, 2005 , 72 - 93

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

[12] I. Baumgart, B. Heep, and S. Krause, “OverSim: A flexible overlay network simulation framework,” in Proceedings of 10th IEEE GlobalInternet Symposium (GI ’07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA, May 2007, pp. 79–84.

[13] Ingmar Baumgart, Bernhard Heep, Stephan Krause , “OverSim: A scalable and flexible overlay framework for simulation and real network applications “,IEEE Xplore Conference : Peer-to-Peer Computing , 2009 .

[14] Vikash Jaglan , Dr. Sukhvir Singh , Advanced Chord Algoritthm , International Journal For Research IN Applied Sciense And Engineering Technplpgy (IJRSSET) . Vol. 1, Issue II, September 2013, ISSN: 2321-9653

[15] Sonia , Mr.Jiltender Arora , Improved Chord Algoritthm , International Journal For Research IN Applied Sciense And Engineering Technplpgy (IJRSSET) . Vol. 2, Issue XII, December 2014, ISSN: 2321-9653