Abstract—Chord has been widely used as a routing protocol in structured peer-to-peer overlay networks, but there is a lot of redundant information in the node finger table. This paper proposes a strategy that using the node's counterclockwise node information to replace the redundant information, the strategy uses the space effectively which occupied by the redundant information. This paper also proposes a strategy to solve the problem that repeated search in p2p network by establishing an objective resource table. Then this paper designs an advanced chord routing algorithm based on redundant information replaced and objective resource table (AChord). Theoretical and experimental analysis show that AChord can improve search performance effectively, and it also proposes a new way of thinking for the structured P2P network search.

Keywords-Chord; redundant information; finger table; objective resource table; AChord

I. INTRODUCTION

Peer-to-Peer (P2P) is a distributed networks, and the participants share their own hardware and software resource such as processing power, storage capacity, network connectivity, printer [1,2], etc. Due to their scalability, fault-tolerance and self-organization, P2P technology is widely used in a variety of services. It brings some convenience to people, but it also brings a lot of problems which can not be ignored such as routing, information retrieval bandwidth intellectual property and security problems [3]. Among the problems which the search problem is one of the most critical issues that the application of P2P networks have to face. Currently there may be long detention; low resource hit rate and excessive bandwidth depletion [4], etc. According to the way of node organization, existing P2P networks is divided into structured and unstructured P2P networks. So P2P search algorithms are also divided into structured and unstructured P2P search algorithm. In unstructured networks such as Gnutella, Gnutella2, Freenet [5, 6], etc, flooding algorithm is been adopted which uses TTL to control search, but beyond a certain range, the search will not be further expanded upon, and it brings a lot of redundant messages, wasted network bandwidth and other problems. Structured networks such as Tapestry, Pastry, and Chord are based on DHT (Distributed Hash Table) technology [3, 6], so the search has certainty and scalability. But as the network size gradually increased, structured P2P network search also appears too much messages and long search delay problems. This paper further studies Chord search algorithms and analyzes their advantages and disadvantages, and presents an advanced chord routing algorithm based on redundant information replaced and objective resource table (AChord). Theoretical and experimental results show that AChord is feasible and efficient.

II. RELATED WORK

A. Chord protocol and the routing algorithm

Chord is proposed by MIT as a resource routing protocol based on DHT [7, 8]. Chord uses consistent hash. It allocates each node and resource an m bit identifier, the nodes and resource will be mapped to a logical ring whose size is $2^m$, the ring is known as the Chord ring. When searching files, the index of each file is represented as (K, V). K represents a file name or hash value of file attributes, and V represents an IP address which actually stores K. All files’ indexes compose a large file index table, and just enter K value of the search objective file, you can find out node address which stores the file in this table.

Chord allocates its identifier space in clockwise and the order from small to large. Similarly, each resource also has a unique m-bit identifier KeyID. The resource belongs to the first node whose identifier id is equal to or greater than KeyID in the identifier space numerically (clockwise Chord ring) .This node is called successor node of the resource and it is responsible for storing the resource index. This node and resources are mapped to the ring which has the same range [0--$2^m$-1]. Therefore, resource location becomes a simple positioning of its successor nodes.

For example, in Fig.1, there is a Chord ring whose m is equal to 6 (m is the bit size of the keywords or identifiers). The identifier ranges from 0 to 63($2^6$), a total of 64 (2^6). There are 15 nodes on the Chord ring and the resource is assigned to its successor node on the Chord ring according to its own key value. The node uses [(id + 2 ^(-i)) mod 2^m] to get their successors, such as the first pointer of node N8 is (8+2^(-1))mod 2^6=9, the second is (8+2^(-2))mod 2^6=10, the third is (8+2^(-3))mod 2^6=12, so their successor nodes are N14. The last is (8+2^(-4))mod 2^6=40, and its successor node is N42. That is the resources whose key ranges from 9 to 14 are assigned to their successor N14, the resources whose key ranges from 39 to 42 are assigned to their successor N42 and so on.

978-1-4244-5539-3/10/$26.00 ©2010 IEEE
For example, considering node 8 wants to find key 53. Firstly, node 8 searches if key 53 is in itself, if in itself, the search is successful, otherwise, node 8 searches the node that is less than or equal to key 53 and the closest to key 53 in its finger table, the result is node 42. Node 8 will send the query to node 42. Then node 42 finds that node 51 is less than or equal to key 53 and is the closest to key 53 in its finger table. Then Node 42 will send the query to node 51. Node 51 finds node 56 in its finger table, and there is no other node from node 51 to node 56, so Node 51 will send the query to node 56. Finally, node 56 discovers key 53 and sends key 53 to node 8, this query is successful. The basic Chord lookup method which Node N search resources K is as follows:

Step1: Node N requests resource K, it searches in itself firstly, if the search is successful, return. Otherwise, go to Step2.

Step2: Search in the finger table of current node, find the node N’ which is less than or equal to K and the closest to K, if find out N’, send the search message to N’, repeat step2 until the search is successful, else the first successor node of the current node’s finger table is object node, return.

B. The existing and improving Chord routing algorithm

There are two problems in the original Chord. One is that there is redundant information in node’s finger table, such as N8 has three pointers point to N14 in Fig.1. The other is that Chord ring is constructed by consistent hash, its logical address and physical address may vary greatly, so the search may appear a "detour" phenomenon, which also affects the search performance of Chord. Domestic and foreign researchers have proposed some strategy to improve the search performance of Chord. Stanford University proposed a bilateral Chord [9], which adds counterclockwise node in the finger table, so the lookup method can search from both clockwise and counterclockwise, it improves the search performance, but it occupies much more space. MIT has proposed a method must maintains much more routing tables [12].

This paper designs an AChord routing algorithm, which deletes the redundant information in the node’s finger table and adds counterclockwise node information, objective resource table, which has a very good good compromise both in space complexity and time complexity.

III. ACHORD LOOKUP METHOD BASED ON REDUNDANT INFORMATION REPLACED AND OBJECTIVE RESOURCE TABLE

The redundant information in the finger table of Chord ring not only takes up valuable space, but also increases search delay. Therefore, it is necessary to solve this problem. There is a direct idea that using inspired information to replace the redundant information, so that each node can use the space effectively and establish links with much more nodes, the idea can improve the search performance greatly. Considering that a search may be repeated many times in Chord ring, for example, N8 sends 100 times search requests for getting K53 in ten minutes, it must repeat N8-> N42-> N51-> N56 routing 100 times in the Chord ring. The original Chord lookup method not only increases the message in the network and takes up valuable bandwidth resources, but also increases search delay, therefore it is a critical question to solve the problem that repeated search in the Chord ring.

A. The strategy for redundant information replaced in the finger table

It can be seen from the structure of the finger table, the identifier of the successor nodes increase clockwise in the Chord ring, and each node only gets half ring’s information from node’s clockwise, however, if the objective resource exists in the predecessor of the node or near counter-clockwise distance of the node, there will be a great delay if using the original Chord lookup method. So this paper presents a strategy of improving finger table to solve the problem. The specific strategy of establishing node’s finger table is as follows: firstly compute the successor in clockwise by using \([id \times 2^{i-1}) \mod 2^m\], if the node which the current pointer points to is equal to the former pointer points to, compute the successor in counter-clockwise by using \([2^m + (id-2^i) \mod 2^m]\). Similarly, redundant information will not be found in counter-clockwise. In order to keep the clockwise order of the node in the finger table, we make the counter-clockwise address and node in the trail of the finger table, and the counter-clockwise address is from low to high. The strategy can get the counter-clockwise information of the node to speed up search. It also saves valuable space than the bilateral Chord. The original and improved finger tables of N8 in fig. 1 are as follows:
TABLE I. THE ORIGINAL (A) AND IMPROVED (B) FINGER TABLES OF N8

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>N8+1</td>
<td>N14</td>
</tr>
<tr>
<td>N8+2</td>
<td>N14</td>
</tr>
<tr>
<td>N8+4</td>
<td>N14</td>
</tr>
<tr>
<td>N8+8</td>
<td>N18</td>
</tr>
<tr>
<td>N8+16</td>
<td>N27</td>
</tr>
<tr>
<td>N8+32</td>
<td>N42</td>
</tr>
<tr>
<td>N8+16</td>
<td>N27</td>
</tr>
<tr>
<td>N8+8</td>
<td>N18</td>
</tr>
<tr>
<td>N8+1</td>
<td>N14</td>
</tr>
</tbody>
</table>

For example, the objective resource which N8 requests exists in N63 in Fig. 1, if N8 adopts the original finger table, it will cost many hops, but if uses the improved finger table, it only costs one hop. It shows that the improved finger table has a good search performance.

B. The establishment of the objective resource node table

In the Chord ring, some node often gets resource from another node, and there may be many hops between the two nodes, so it reduces the search performance. To solve this problem, this paper proposes to establish the objective resource node table. The so-called objective resource table is that stores the objective resource in the source node. Each node has an objective resource table whose size is half of the finger table. This paper uses LRU algorithm to manage the objective resource node table. The so-called objective resource table is that stores the objective resource in the source node. The table includes two items, one is the objective node, and the other is its direct predecessor. The structure of the table is as follows in tab. 2.

TABLE II. THE STRUCTURE OF OBJECTIVE NODE TABLE

<table>
<thead>
<tr>
<th>predecessor</th>
<th>objective node</th>
</tr>
</thead>
<tbody>
<tr>
<td>N18, N21</td>
<td>(N18, N21)</td>
</tr>
<tr>
<td>N32, N35</td>
<td>(N32, N35)</td>
</tr>
<tr>
<td>N48, N51</td>
<td>(N48, N51)</td>
</tr>
</tbody>
</table>

As is shown in the tab. 2, the first item is the objective node’s predecessor, and the second is the objective node. This structure of the table is good and useful for positioning the requested resource quickly. According to the node’s predecessor and itself, it is easy to know the range of resource identifier that the node has, so it is fast to find the requested node. For example, if N1 often gets resource from N21, N35, N51 in fig. 1, we will add (N18, N21), (N32, N35), (N48, N51) to N1’s objective resource table. From the objective resource table, we know that N21 has resource whose identifier belongs to [19, 21], N35 has resource whose identifier belongs to [33, 35], N51 has resource whose identifier belongs to [49, 51]. When N1 requests resource K49, N1 searches it in its objective table firstly and can easily find N51. It costs one hop and finishes the search. So it can improve the search performance.

C. New node joining and leaving

When a new node M wants to join the Chord ring, it should be guided by a node M’ in the ring. Node M creates its own finger table by using the strategy in the A section, and notifies the other nodes in the ring to update their finger tables and objective resource table. When the new node M completed the finger table setup and updated the routing information, some information in the new node’s successor will be transferred to the new node. For example, in Fig. 1, the K-value which 42 < K <= 48 are stored in Node 48, when a new node 45 joining the Chord ring, K-value which 42 < K <= 45 are stored in Node 45, 45 < K <= 48 are stored in Node 48. When a node m leaving the Chord ring normally, it will transfer the K-values stored in it to its successor node, and notify the predecessor that it is leaving the network, the predecessor will update its successor to point to the node m’ successor, and it should update the objective resource table.

D. AChord routing algorithm

The theoretical analysis shows that the strategies of redundant information replaced and the establishment of the objective resource table can improve the search effectively. Compared with the existed Chord routing algorithms, AChord has a very good compromise both in space complexity and time complexity. The AChord routing algorithm is as follows:

```
// Node N request resource K
Step1: Node N requests resource K and it search in the local firstly, if successful, return. Otherwise, go to Step2.
Step2: Search the objective resource table of current node, if the search is successful, updating the objective resource table. Otherwise, go to Step3;
Step3: Search the finger table of current node, and find out the node N’ that greater than or equal to K and the closest to K. If find the node N’, it will send the inquiry request to N’ and go to Step2, else the first successor node in current node’s finger table is object node, return.
```

IV. EXPERIMENTS AND DISCUSSION

P2Psim is a free, multithreaded, discrete event-driven simulator to evaluate, investigate, and explore P2P protocols designed by MIT PDOS Lab. In this paper, experiments are done under the development environment of a desktop computer with 1G memory, 2.6GHz Intel CPU with Red Hat 9.0 Operating System and GCC4.2 compile tool for algorithm performance assessment. This paper compares with the Chord and AChord routing algorithms by the num of hops and the delay time, the number of experimental nodes between 1000 and 10000, and increased 1000 node each time, we do many times of each group experimental data to calculate the average value, the results are shown in Fig. 2, Fig. 3.

Figure 2. The average lookup - hops comparison
Fig. 2 shows AChord routing algorithm is superior to Chord under the same network conditions. As it can be seen from Fig 2, the number of hops which Chord routing algorithm costs increases quickly with the increasing size of network nodes. Because the average path length of Chord is about $1/2 \log_2 N$ and the AChord are about $1/2 \log_2 (N/2)$, and because of the increase of the objective resource table, some search can be completed only in one hop in AChord, so the average hops of AChord routing algorithm is better than Chord.

![Figure 3. The average lookup -delay time comparison](image)

The number of nodes ($*10^3$)

Fig. 3 shows that the search delay time of Chord and AChord under the same conditions. It can be seen in Fig 3 that the average search delay of Chord is greater than the AChord, because Chord searches only clockwise while AChord searches both clockwise and counterclockwise, and because of the increase of the objective resource table, some search can be completed only in one hop in AChord, so the average delay time of AChord routing algorithm is better than Chord.

V. CONCLUSION AND FUTURE WORK

This paper firstly studies the Chord routing algorithm and some improvement strategies, proposes a strategy that using the node’s counterclockwise node information to replace the redundant information. The strategy uses the space effectively which occupied by the redundant information. Secondly we establish an objective resource table to solve the repeated search problem. Finally we design the AChord lookup method. AChord has a very good compromise both in space complexity and time complexity. Our future work will study Chord routing algorithm to solve the "detour" problem, we hope the search performance of Chord routing algorithm will be further improved.

ACKNOWLEDGMENT

The subject is sponsored by the Science and Technology Support Program of Jiangsu Province (No. BE2009158), the Natural Science Fund of Higher Education of Jiangsu Province (No. 09KJB520010), Ph.D. Programs Foundation of the Ministry of Education of China (NO. 20093223120001), Special Fund for Fast Sharing of Science Paper in Net Era by CSTD (NO. 2009117). The authors would like to thank the anonymous referees for the useful suggestions for improving this paper.

REFERENCE