# The Design and Implementation of SS-Chord Searching Algorithm

**Qin Xue**

*Department of Information Technology, Hubei University of Police, Hubei Wuhan , China*

*superxueqin@163.com*

**Abstract:** Chord searching algorithm is lack of the consideration of node function difference and has the detouring problem. Aimed at all the above problems, subnetwork and super node are applied to the Chord and then we get SS-Chord searching algorithm. The design thought, network model and resource location process of SS-Chord searching algorithm are expounded in detail. Finally, from the view of time complexity and space complexity, the two searching algorithms are analyzed comparatively.The simulated experiment shows that SS-Chord searching algorithm is better than Chord searching algorithm in the overall performance.

**Keywords:** algorithm theory; searching algorithm; simulated experiment; peer to peer; super node

# SS-Chord 搜索算法的设计与实现

**薛琴**

湖北警官学院信息技术系，湖北武汉，中国，430034

superxueqin@163.com

【摘要】针对 Chord 搜索算法存在对节点性能差异性考虑不周和绕路问题，将子网和超级节点应用到 Chord 后得到 SS-Chord 搜索算法。详细阐述了 SS-Chord 搜索算法的设计思想、网络模型以及资源定位过程，并从时间复杂度和空间复杂度两个方面对两种搜索算法进行了对比分析，仿真实验表明在总体性能上 SS-Chord 搜索算法比 Chord 搜索算法更优。

【关键词】算法理论；搜索算法；仿真实验；对等网；超级节点

## 1 引言

在对等网 P2P(Peer to Peer)的搜索算法中，集中目录式搜索算法存在中央服务器的系统瓶颈问题[1]，洪泛搜索算法的可扩展性较差[2]，因此，基于分布式哈希表 DHT（Distributed Hash Table）的算法是目前研究的热点。而基于分布式哈希表算法的典型代表——Chord 搜索算法以其高稳定性[3]，在节点动态变化频繁的对等网中尤为引人注目。然而，在对等网的资源定位过程中，Chord 搜索算法对节点的性能差异考虑不足，并且存在绕路问题。因此，在 Chord 基础之上设计并实现一种新的搜索算法势在必行。

## 2 Chord 搜索算法的不足之处

### 2.1 绕路问题

由于 Chord 搜索算法中的相邻节点是通过虚拟地址定义的[4]，虚拟地址和物理地址之间没有必然联系，导致实际的路由过程中存在“舍近求远”的隐患，即网络存在绕路问题。覆盖网络上的两个邻近节点理论上看似距离很近，但实际上要想找到对方却要经过很长的路径，使得基于这种覆盖网络所做的评测是不可靠的；相反，实际距离很近的两个节点，因为被映射到了覆盖网络上距离很远的位置，却要在网络中绕很长的距离才能找到对方，导致大量重复路径存在，增加网络流量，浪费带宽[5]。

### 2.2 没有充分考虑节点的性能差异

Chord 搜索算法没有考虑网络中各个的节点的性能（包括节点存储能力、计算能力、网络带宽等）差异[6]，所有节点的地位都是平等的，所有节点都被赋予同一责任，这将导致高性能的节点的能力没有被充分挖掘出来，而相对性能较差的节点却担负着繁重的任务，使网络的稳定性和搜索信息的速度受到了严重影响。

## 3 SS-Chord 搜索算法的设计与实现

在 Chord 搜索算法的基础上，考虑节点的实际物理地址和节点性能的差异，提出一种基于子网（Subnetwork）和超级节点（Supernode）的 SS-Chord 搜索算法。

### 3.1 SS-Chord 搜索算法的设计思想及网络模型

把在实际物理网络中邻近的节点组成一个子网，此子网资源定位算法遵循 Chord 搜索算法，这样所有的节点可以组成若干个子网，然后把这些子网再互联起来构成主干网。把网络中一部分高性能的节点设为超级节点，在进行资源定位时通过超级节点来进行。每个子网中可以选择一个超级节点，所有的超级节点可以构成一个 Chord 环，这个 Chord 环即为主干网。这样整个网络可看成由若干个 Chord 型子网构成的 Chord 环。

基于 SS-Chord 搜索算法的网络模型图如图 1 所示，其中 SN 代表超级节点。

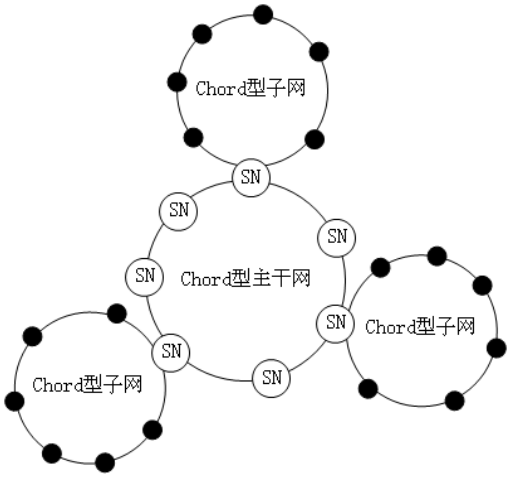


**图 1 基于 SS-Chord 搜索算法的网络模型图**

根据节点能力的不同，SS-Chord 的节点分为四类：①超级节点，即子网和主干网交接的节点，由一个子网内具有很好的计算能力、存储能力和网络带宽的高性能节点充当；②备份节点，由性能较强的节点充当，且处于子网中，主要用于备份超级节点的数据，当超级节点失效时，代替超级节点的角色；③普通节点，即子网中除超级节点和备份节点以外的所有节点，由性能一般的节点充当；④管理节点，即由子中专门的终端来充当，它们具有公开的身份，不具备子网中普通节点所具备的功能，其作用是管理并监视子网节点，以及运行超级节点的选择算法。

### 3.2 SS-Chord 搜索算法的资源定位过程

SS-Chord 搜索算法中节点 n 查找数据对象的算法用伪代码表示如下：

```
//key 为所查找数据对象通过哈希函数得到的 64 位关键值，低 32 位用 key1 表示，高 32 位用 key2 表示
Procedure location(n,key)
{   if(n==supernode)
        SuperRoute(n,key2);
    else if (n==normalnode||n==backupnode)
        SubnetworkRoute(n,key1);
        else   {cout<< "Node n Error!"<<endl;
            return NULL;   }
}
SuperRoute(n,key2)
{   if (look up cache(n,key))
     {   Send Query Result( );
         return n;   }
//如果在超级节点的查询缓冲区有与 Key 匹配的索引记录，立即返回结果给查询节点
     else                    //否则在主干网上查询数据对象
        { n'=n;
          while(key2 ∉( n',n'.successor))
           n'=Superclosest_preceding_finger (n',key2);
          return n'.successor;
          update Cache( );   }
 }
//在节点 n 的路由表中，从后往前找到与 K 最接近且小于 K 的节点
Superclosest_preceding_finger (n,key)
{   for (i=32;i>=1;i--)
    if (super_finger[i].node∈(n,key))
        return super_finger[i].node;
    return n;   }
SubnetworkRoute(n,key1)
{   while（TTL!=0）//TTL 为查询数据对象可允许的次数
{   n'=n;     while(key1 ∉（ n',n'.successor ））
```


978-1-935068-01-3 © 2009 SciRes.

```
        n'=closest_preceding_finger (n',key1);
        return n'.successor;
        update Cache( );    }
    SuperRoute(n,key2);     }
```

## 4 SS-Chord 搜索算法的性能分析

### 4.1 空间复杂度

Chord 搜索算法中的每个节点都保存有一个路由表，此表最多 m 项，m 为标识符的位数，一般为 32-160bit，设在对等网的节点饱和时的规模为 N，则 $m=\log_2 N$，因此 Chord 搜索算法的空间复杂度是 $O(\log_2 N)$，Chord 搜索算法中所有节点的路由表的项数为 $R1=N\times(\log_2 N)$。

SS-Chord 搜索算法中的普通节点和备份节点只保存有一个路由表，此路由表只负责子网内的资源搜索。假设对等网的节点饱和时的规模为 N，子网内节点个数都为 L(为便于分析，取一种子网内节点个数都相同的状态)，则子网个数为 INT(N/L)，其中 INT 表示取整数的含义，此时超级节点个数也为 INT(N/L)。此时，子网内的普通节点和备份节点的空间复杂度为 $O(\log_2 L)$。SS-Chord 搜索算法中的超级节点被哈希后的值分为两部分，低 32 位用于子网内资源搜索，高 32 位用于主干网资源搜索，所以超级节点有 2 个路由表。此时，超级节点的空间复杂度为 $O(\log_2 L+ \log_2 INT(N/L))$。此外，由于 SS-Chord 搜索算法中的超级节点有缓存技术，所以超级节点的空间复杂度中还包括缓冲区的 buffer 大小，考虑到缓存内容并非数据对象，而是为数据对象的索引，一般占用空间不需太大，2MB 以内即可，这相对于超级节点的存储空间几乎可以忽略不计。因此，SS-Chord 搜索算法中所有节点的路由表的项数为 $R2= INT(N/L)\times(L-1)\times(\log_2 L)+INT(N/L)\times(\log_2 L+\log_2 INT(N/L)) = INT(N/L)\times(L\times\log_2 L+ \log_2 INT(N/L)) = INT(N/L)\times(\log_2(L^L\times INT(N/L)))\leqslant(N/L)\times\log_2(L^{L-1}\times N)$。

因为 $0<L\leqslant N$ 且 $L\geqslant 1$ 成立，所以 $0<L^{L-1}\leqslant N^{L-1}$ 成立，即 $\log_2 L^{L-1}\leqslant\log_2 N^{L-1}$ 成立，则 $\log_2 L^{L-1}+ \log_2 N\leqslant \log_2 N^{L-1}+ \log_2 N$ 成立，即 $\log_2 (L^{L-1}\times N)\leqslant\log_2 (N^{L-1}\times N)$ 成立，则 $(N/L)\times\log_2 (L^{L-1}\times N)\leqslant(N/L)\times\log_2 (N^{L-1}\times N)$ 成立，即 $(N/L)\times\log_2 (L^{L-1}\times N)\leqslant N\times\log_2 N$ 成立，所以 R2<R1 恒成立。

因此，SS-Chord 搜索算法中所有节点的路由表项数比 Chord 搜索算法中所有节点的路由表项数少，从空间复杂度角度看，SS-Chord 搜索算法比 Chord 搜索算法要优。

### 4.2 时间复杂度

#### 4.2.1 资源定位的跳数

考察资源定位跳数，应该从最坏的情况出发。当 Chord 搜索算法中节点饱和时应该是最坏的情况，此时进行资源定位的跳数应该是上限值。设 Chord 中节点饱和时的节点个数为 N，则查询跳数为 $\log_2 N$，即 Chord 搜索算法中资源定位的跳数最多为 $\log_2 N$，是一个定值。

在 SS-Chord 搜索算法中，整个网络模型由主干网和子网构成，为便于分析，假设总共节点数为 N，子网内节点个数都为 L，则子网个数为 t，t=INT（N/L）。由于每个子网内都有一个超级节点，所以超级节点个数也为 t。同样在 SS-Chord 搜索算法中考察资源定位跳数也从最坏的情况出发。假设子网查询成功率为 p，超级节点查询缓冲成功率为 q。最坏的情况是首先在子网内查询失败，然后由超级节点查询缓冲也失败，最后在主干网上由超级节点查询成功时，此时资源定位的跳数 $Y=\log_2 L\times(1-p)+ \log_2 t\times(1-q)\times(1-p)$。当子网查询成功率为 p 和超级节点查询缓冲成功率为 q 增大时，Y 减小。极限情况时 p 和 q 趋近于 1，此时 Y 趋近于 0，这是最好的情况。

因此 SS-Chord 搜索算法具有很强的可扩展性，随着网络规模的增大，p 和 q 减少查询跳数的作用越明显。

#### 4.2.2 平均每一跳的资源定位延迟

Chord 搜索算法中没有考虑覆盖网络和底层网络的差异，资源定位的每一跳是在覆盖网络上进行的，实际上覆盖网络上的一跳有可能需要在底层网络中进行多跳路由，因此，平均每一跳的资源定位延迟很大，从而降低了资源定位的效率。

SS-Chord 搜索算法的一个优势是充分考虑了覆盖网络和底层网络的差异，在物理位置上临近的节点处于同一个子网，资源定位的一跳在底层网络中也是一跳，因此平均每一跳的资源定位延迟较小，从而大大提高节点间的访问和连接速度。

因此，SS-Chord 搜索算法比 Chord 搜索算法平均每一跳的资源定位延迟要小得多。

#### 4.2.3 数据的传输效率

Chord 搜索算法中每次数据传输，可能都要路由到实际物理位置较远的节点。而 SS-Chord 搜索算法中，资源搜索时都先在本地子网中进行，如果在本地子网中找到了资源，就直接在本地下载，因此数据传输速度快，效率高。此外，SS-Chord 搜索算法中的超级节点提供了缓存机制，把当前网络中搜索较多的数据索

引放在缓冲区，使得网络中很大一部分查询可以在子网内完成，减少了网络流量，提高了数据传输效率。因此，SS-Chord 搜索算法比 Chord 搜索算法的传输效率要高得多。

通过以上分析比较得知，在时间复杂度和空间复杂度上 SS-Chord 搜索算法比 Chord 搜索算法都具有明显的优势，值得推广。

## 5 仿真实验

在 Linux 操作系统下通过 P2Psim 模拟器做仿真得到结果如下。

### 5.1 资源定位的开销

比较 Chord 搜索算法和 SS-chord 搜索算法的资源定位开销的方法是，当网络规模从 100 个节点增加到 1000 个节点时，网络中节点个数和平均定位路由长度关系图如图 2 所示。

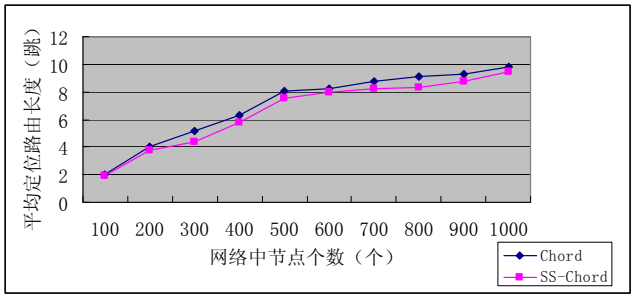


**图 2 网络中节点个数和平均定位路由长度关系图**

图 2 表明，在网络中节点个数相同时，SS-Chord 搜索算法比 Chord 搜索算法平均定位路由长度要小，这是 SS-Chord 搜索算法考虑覆盖网络和实际物理网络的差异性划分子网的结果。

### 5.2 资源定位访问延迟

在进行资源定位时，覆盖网络和底层网络的实际距离不一致性可以通过资源定位访问延迟来反映。访问延迟越大，覆盖网络和底层网络的差异越大；反之，差异越小。当网络规模从 100 个节点增加到 1000 个节点时，网络中节点个数和资源定位访问延迟关系图如图 3 所示。

图 3 表明，随着网络中节点个数的增加，Chord 搜索算法资源定位延迟增长较快，SS-Chord 搜索算法

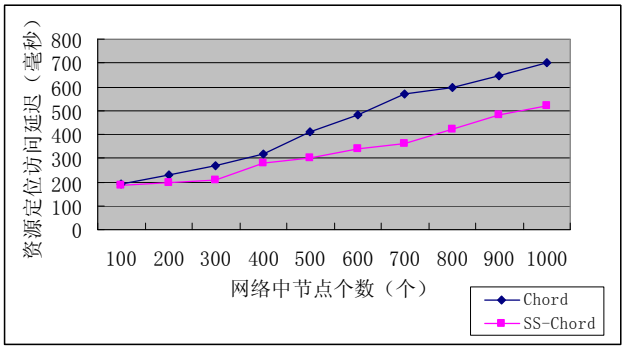


**图 3 网络中节点个数和资源定位访问延迟关系图**

资源定位延迟增长较慢，这是 SS-Chord 考虑了节点的实际物理地址，划分子网的结果。

## 6 结束语

针对 Chord 搜索算法的绕路问题和没有充分考虑节点的性能差异性问题，将子网和超级节点应用到 Chord 后，提出一种基于分布式哈希表的 SS-Chord 搜索算法。该算法的空间复杂度和时间复杂度比 Chord 搜索算法都要小，并且仿真实验表明，随着网络中节点个数的增加，在资源定位开销和资源定位访问延迟方面，SS-Chord 搜索算法比 Chord 搜索算法具有显著优势，值得推广。

## 致谢

真诚地感谢华中科技大学计算机学院肖道举老师给论文提出了宝贵的修改意见。

## References（参考文献）

[1] H Liu,P Luo,Z Zeng.A structured hierarchical P2P model based on a rigorous binary tree code algorithm[C].Future Generation Computer Systems,23 June 2006.

[2] Yejiang Zhang, Zhitang Li, Zhengbing Hu, et al. A P2P E-commerce Related Network Security Issue: P2P Worm[J]. Electronic Commerce and Security, 2008 International Symposium on 3-5 Aug. 2008:114-117.

[3] Zhu Y,Hu Y.Efficient proximity-aware load balancing for DHT-based P2P systems[J].IEEE Trans on Parallel and Distributed Systems,2005,16(4):349-361.

[4] Zou Fu-tai,Pan Le-yun,Wu Zeng-de,et al.Structured peer-to-peer topology model based on session heterogeneity[J].Journal of Shanghai Jiaotong University,2004,38(Sup):145-147.

[5] Ji ping,XiongYouwei,Jinsheng Li,et al.Chord:Ipv6 Based Topology-Aware Chord[J].Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005 23-28 Oct, 2005:4-4.

[6] Hung Nguyen Chan, Van K.N, Giang Ngo Hoang.Characterizing Chord, Kelips and Tapestry algorithms in P2P streaming applications over wireless network[J].Communications and Electronics, 2008. ICCE 2008. Second International Conference on 4-6 June 2008:126-131.