一种基于双标识符的 Chord 路由模型①

王必晴，钟志水，孟伟东，袁晓勇，王福成

(铜陵学院 数学与计算机科学系, 铜陵 244000)

摘 要：针对 Chord 协议的路由表只能覆盖一半标识符空间的问题，提出了一种基于双标识符的 Chord 路由模型。该模型除了按照 Chord 协议给每个节点和关键字分配一个顺时针标识符，另外还分配一个逆时针标识符。这样，一个 Chord 环上的节点或待查找的关键字便拥有双标识符。因此，每个节点能构造顺时针和逆时针两张路由表，可以覆盖整个标识符空间。理论分析和仿真实验表明，改进的 Chord 路由模型减少了平均查找跳数，提高了路由效率。

关键词：Chord 协议；标识符；路由；算法

Routing Model for Chord Based on Double Identifier

WANG Bi-Qing, ZHONG Zhi-Shui, MENG Wei-Dong, YUAN Xiao-Yong, WANG Fu-Cheng

(Department of Mathematics and Computer Science, Tongling University, Tongling 244000, China)

Abstract: A routing model for chord based on double identifier is proposed to address the problem that the routing table in Chord only covers half of the identifier space. The model assigns each node or key not only a clockwise identifier according to Chord but also a anticlockwise identifier in addition. So each node or key has double identifiers. Thus each node maintains two routing tables:clockwise routing table and anticlockwise routing table. Performance analysis and simulation experiments show that improved Chord routing model reduces the average lookup path length and gets higher efficiency.

Key words: chord protocol; identifier; routing; algorithm

基于分布式哈希表的结构化 P2P 网络 Chord[1]，由于具有结构简单、查找准确、负载平衡等特点，正日益成为 P2P 网络研究和应用的热点。在经典的 Chord 算法中，节点维护的路由表只能覆盖一半标识符空间，见图 1。显然，只要目标节点落入路由表没有覆盖的半环，就必须通过至少一个中间节点才能找到，影响了查找效率。

目前，有很多文献[2-6]对 Chord 做了进一步的研究。文献[2,3]提出了 One-Hop 的算法，其基本构想是在节点中存储全部节点的路由信息，那么所有的查找有可能在常数跳内完成。文献[4]增加了路由信息的存储，系统将节点分成 k 组，每个节点路由表保存了此节点所在组内的所有节点信息和组外的其他 k-1 个节点信息，并且通过 gossip 的方式进行更新。文献[5]引入了反应式路由维护策略：节点在查询的时候，通过其他节点的回复信息更新本地的路由信息。文献[6]提出了一系列基于斐波那契数列的 chord 算法，其网络直径为 0.72log2N，平均的路径长度为 0.398log2N。

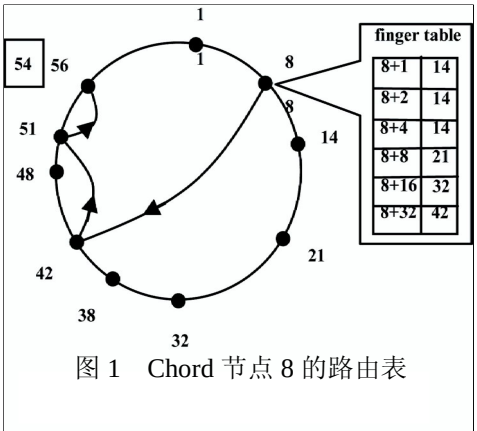

图 1 Chord 节点 8 的路由表

① 基金项目:安徽省高等学校省级自然科学研究项目(KJ2011Z377);安徽省高等学校省级优秀青年人才基金(2011SORL145);铜陵学院 2009 年度院级科研项目(2009tlxy22)

收稿时间:2011-11-14;收到修改稿时间:2011-12-29

本文在以上研究的基础上，提出了一种基于双标识符的 Chord 路由模型。该模型除了按照 Chord 协议给每个节点和关键字分配一个顺时针标识符，另外还分配一个逆时针标识符。这样，一个 Chord 环上的节点或待查找的关键字便拥有双标识符。因此，每个节点能构造顺时针和逆时针两张路由表，可以覆盖整个标识符空间。通过这样的方式，改进的 Chord 路由模型减少了平均查找跳数，提高了路由效率。

1 基于双标识符的Chord路由模型

为了更清楚地描述基于双标识符的 Chord 路由模型，下面首先定义一些相关的概念。

1.1 基本概念

如果没有特别说明，文献[1]中的所有定义和记法均适合本文。在此基础上，再给出以下定义。

定义 1 顺时针标识符 按照 Chord 协议为每个节点和关键字分配的 m 比特的标识符，分别记为 n 和 k。

定义 2 逆时针标识符 为每个节点和关键字分配的 m 比特的标识符，其计算方法为 $(2^m-n)\bmod 2^m$ 和 $(2^m-k)\bmod 2^m$，分别记为 n_0 和 k_0。

定义 3 顺时针路由表 每个节点按照 Chord 协议维护的路由表。

定义 4 逆时针路由表 每个节点维护的有 m 个表项的路由表，其中第 i 个表项为$((2^m-n)\bmod 2^m)$。finger[i].node。

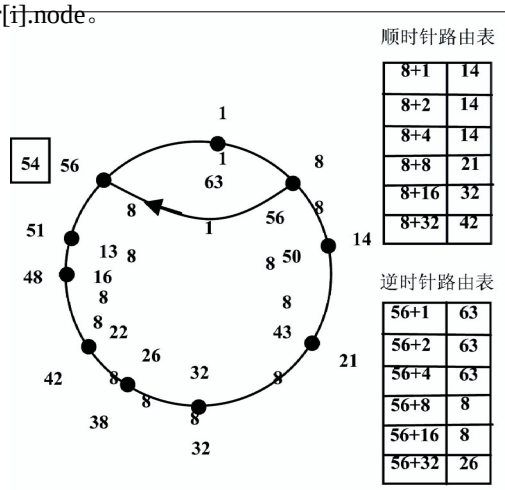

图 2 改进模型的路由算法示例

例如，图 2 给出了一个具有双标识符的 Chord 环以及节点 8 的顺时针和逆时针路由表。其中，环外为顺时针标识符，环内为逆时针标识符。可以看到，两张路由表覆盖了整个标识符空间。

1.2 路由方向选择算法

当一个节点收到查找关键字 k 的请求时，首先要选择从哪个方向进行路由。

假设 n 是当前发起查找的节点，它要查找关键字 k 的后继节点 successor(k)，则 n 与 k 之间顺时针方向环长为 $(k-n+2^m)\bmod 2^m$，逆时针方向的环长为 $(k_0-n_0+2^m)\bmod 2^m=(n-k+2^m)\bmod 2^m$。显然，选择环长较小的方向进行路由，跳数更少，效率更高。例如，图 2 中节点 8 查找关键字 54 的后继结点，其顺时针方向的环长为$(54-8+2^6)\bmod 2^6=46$，而其逆时针方向的环长为$(8-54+2^6)\bmod 2^6=18$，由于逆时针方向的环长小于顺时针方向的环长，所以应选择逆时针方向进行路由。

路由方向选择算法的伪代码如下：

```
//路由方向的选择算法
n.choose_direction(k)
clockwise_length=(k-n+2^m)mod2^m;
anticlockwise_length=(n-k+2^m)mod2^m;
direction= minimum(clockwise_length,
   anticlockwise_length);
if(direction= =clockwise_length)
   return n.clockwisefind_successor(k);
else
   return n.anticlockwisefind_successor(k);
```

1.3 逆时针路由算法

顺时针方向的路由依然采用初始 Chord 协议的算法，下面介绍逆时针方向的路由算法。假设 n 是当前发起查找的节点，它要查找 k 的后继节点 successor (k)：

(1) 计算 n_0 和 k_0；

(2) 如果 $n_0=k_0$, 则返回 n 的值，查找结束。否则继续执行；

(3) 如果 $k_0\in(n_0, n_0.\text{predecessor})$，则返回 n 的值，查找结束。否则继续执行；

(4) n_0 查找自己的逆时针路由表，找到最大但不超过 k_0 的第一个节点，并将这个查找请求转发给该节点，该节点重复以上查找过程。

需要说明的是，在上述算法的描述中，资源的存放方法和前驱节点的定义仍然遵守初始 Chord 协议的规定。在图 2 中，节点 8 查找关键字 54，根据上一小

节的分析，应选择逆时针方向进行路由，那么只需一跳就可找到 54 的后继节点 56。而从图 1 中可以看到，初始 Chord 需要 3 跳才能完成相同的查找。

逆时针方向路由算法的伪代码如下：

```
//逆时针方向路由算法
n.anticlockwisefind_successor(k)
计算 n0 和 k0;
if(n0= =k0)
  return n;
if(k0∈(n0, n0.predecessor))
  return n;
else n′ = n0.closest_preceding_node(k0);
return n′.anticlockwisefind_successor(k0);
n.closest_preceding_node(k)
for i=m downto 1
  if(finger[i].node∈((n，k))
    return finger[i].node;
```

1.4 性能分析

在改进后的路由模型中，由于每个节点能构造顺时针和逆时针两张路由表，可以覆盖整个标识符空间，所以其路由效率相应会有所提高。当目标节点位于$[n,n+2^{m-1}]$时，由于初始 Chord 环的平均查找跳数为$(\log_2 N)/2$，根据文献[7]，改进模型在$[n,n+2^{m-1}]$半环上的平均查找跳数为$[\log_2(N/2)]/2$。当目标节点位于$(n+2^{m-1},n+2^m-1)$时，根据对称性可知，平均查找跳数也为$[\log_2(N/2)]/2$。由于节点落入两个区域是等概率事件，因此改进模型的平均查找跳数为$\{[\log_2(N/2)]/2+[\log_2(N/2)]/2\}/2=[\log_2(N/2)]/2$，显然比初始 Chord 的平均查找跳数要少。

2 仿真实验

采用 PeerSim 环境，对基于双标识符的 Chord 路由模型进行了仿真实验。依次模拟 500、1000 至 8000 个节点的 Chord 网络，得到每个网络利用改进模型进行路由的平均查找跳数，并与初始 Chord 相对比。在对每个网络的实验中，每个节点对一个随机产生的关键字 k 进行查找，重复 100 次，最后求出所有节点的平均查找跳数，重复 20 次取平均值，得到图 3 所示实验曲线。可以看出，对比 Chord 的路由算法，改进的路由模型明显减少了平均查找跳数，在查找效率上比 Chord 有较大提高，与理论分析的结果一致。

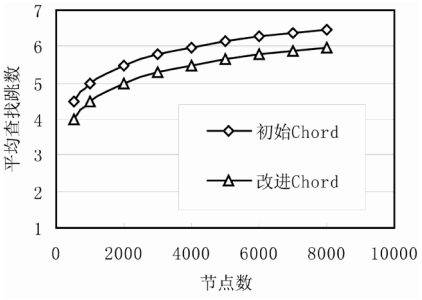

图 3　仿真实验结果

3 结语

本文在 Chord 的基础上，提出了一种基于双标识符的 Chord 路由模型。理论分析和仿真实验均表明，该模型通过扩大路由表覆盖面，减少了平均查找跳数，提高了查找效率。但是逆时针标识符和逆时针路由表的设计会增加一定的系统开销，因此下一步，将继续研究其他提高 P2P 网络查找效率的路由模型和算法。

参考文献

1 Stoica I, Morris R, Karger D, et al. Chord:A Scalable Peer-to-Peer Lookup Service for Internet Application.Proc.of the 2001 ACM SIGCOMM Conference,2001.149–160.

2 Gupta A, Liskov B, Rodrigues R. One Hop Lookups for peer-to-peer overlays.Proc.of the 9th Workshop on Hot Topics in Operating Systems (HotOS- IX).2003.

3 Gupta A, Liskov B, Rodrigues R. Efficient routing for Peer-to-Peer overlays. Proc. of 1st Symposium on Networked Systems Design and Implementation(NSDI'04). San Rancisco, California, 2004.

4 Gupta I, Birman K, Linga P, et al . Kelips:building an efficient and stable P2P DHT through increased memory and background overhead. Procl of 2nd InternationalWorkshop on Peer-to-Peer Systems (IPTP'03).2003 .

5 Leong B, Liskov B, Demaine ED.EpiChord:Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management. Proc. of the 12th International Conference on Networks(ICON). 2004.

6 Cordasco G, Gargano L, Negro A, et al. F-Chord:Improved Uniform Routing on Chord. Networks, 2008.325–332.

7 刘晓锋,吴亚娟,钟乐海.Chord 路由表结构的改进与优化.计算机工程,2007,33(21):102–104.