第 35 卷 第 22 期 计 算 机 工 程 2009 年 11 月
Vol.35 No.22 Computer Engineering November 2009
·网络与通信· 文章编号：1000—3428(2009)22—0117—02 文献标识码：A 中图分类号：TP393

基于物理拓扑的双向搜索 Chord 路由

卢卫青，张振宇，龚红翠，沈庆涛
(新疆大学信息科学与工程学院，乌鲁木齐 830046)

摘 要：Chord 模型未充分利用逆时针上的路由信息，并且未考虑实际网络拓扑结构，使 P2P 系统存在高延迟、低效率的问题。针对该问题，充分利用节点路由表信息和节点在物理网络上的邻近性，并用超级节点存储最近同一簇内的查询结果，提出路由算法 TB_Chord。模拟实验结果表明，该算法在路径长度、访问延迟方面的性能较 Chord 有一定的提高。

关键词：对等网络；Chord 算法；拓扑；双向搜索；超级节点

Physical Topology-based Bidirectional Search Chord Routing

LU Wei-qing, ZHANG Zhen-yu, GONG Hong-cui, SHEN Qing-tao
(School of Information Science and Engineering, Xinjiang University, Urumqi 830046)

【Abstract】 Considering that Chord model does not make good use of the counterclockwise routing information and it does not consider the actual network topology which leads to high latency and low efficiency, this paper presents a routing algorithm, named Topology and Bidirection guided Chord(TB_Chord), which makes full use of the routing information and takes the proximity of peers in underlying physical network into account. It stores the search results in the super-node in each cluster. Experimental results show that this method is superior to original Chord at path length and access latency.

【Key words】 P2P network; Chord algorithm; topology; bidirectional search; super-node

P2P 系统是一种没有中央控制和分层组织的系统，系统中每个节点在功能上都处于同等地位。在大部分的 P2P 系统中如何有效地定位数据仍然是关键问题。

1 问题的提出

Chord[1]是麻省理工学院开发的分散式、结构化 P2P 搜索协议。其路由表具有良好的结构性，查找具有确定性，而且还具有较好的可扩展性。

但 Chord 也存在一定的缺陷[2]，即其路由机制主要是从节点逻辑上的相邻性进行设计，没有考虑网络拓扑的实际结构。在路由中，虽然从源节点到目标节点的路由跳数能有效地控制在某一范围内，但却不能保证每一跳之间距离的合理性。也就是说，在逻辑上相邻的一跳，在物理网络中有可能经过很多跳数甚至好多个自治区域，这样不仅浪费了网络带宽，而且降低了查寻效率。

并且在一般的 Chord 系统中，没有充分利用节点的邻居信息，从它的指针表中可以看出，节点对于顺时针方向邻居节点的信息了解得比较多，而对逆时针上的邻居信息却很少。实际上节点在环上的实际位置分配是随机和均匀的。在这种情况下，如果所查找的关键字在当前节点的前驱位置，也要经过当前节点的最后一个表项中节点的后继，并进行多次转发才可以到达目的节点，这样大大降低了查找效率。

鉴于上述原因，本文在路由过程中充分利用节点路由表信息，结合附近节点的物理位置和拓扑结构，并用超级节点存储最近同一簇内的查询结果，从而有效地缩短路由距离，降低了路由延迟。

2 相关工作

P2P 叠加网络拓扑的构建与底层物理网络不匹配会导致定位资源响应时间增加，为了解决这个问题，研究人员展开了一系列研究[3-5]。文献[3]利用 IP 地址判断 P2P 节点的物理拓扑；文献[4]提出网络相关簇技术，利用 BGP(Bridge Gate Protocol)表的信息将底层拓扑相近的以及在同一自治域下的 P2P 节点连接成簇；文献[5]的方法是使节点自己测量与参考点的往返时间，并根据事先给出的阈值，确定其在叠加层网络中所属的分组。本文假设已利用现有技术将物理网络上相邻近的节点组织成簇。

3 TB_Chord

3.1 双向 Chord 搜索实现

将 Chord 中顺时针的环扩展生成 2 个虚拟环：一个顺时针环(clockwise ring)；一个逆时针环(counterclockwise ring)。

节点存储数据的数据结构由一张长度为 m 的 Finger 表扩展为 2 张长度为 m-1 的 Finger 表，2 张 Finger 表分别负责顺时针查找和逆时针查找各自范围内的节点，如图 1 所示。

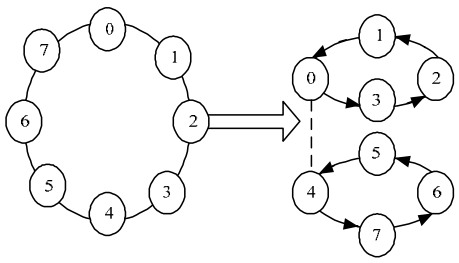

图 1 虚拟环的生成

顺时针 Finger 表(Cw_Finger Table)：是指存储在每个节

作者简介：卢卫青(1982－)，男，硕士研究生，主研方向：对等网络；张振宇，副教授；龚红翠、沈庆涛，硕士研究生

收稿日期：2009-06-23 **E-mail：**luwqxju@sohu.com

点之上的顺时针环的路由表。其结构和 Chord 中的 Finger 表相似，在[(n+1) mod N, (n+2^{m-1}) mod N]范围内，共存储 m-1 条记录，用于存储顺时查找的后续节点信息，划分的区间为[(n+1) mod N, (n+2) mod N), [(n+2) mod N, (n+4) mod N), …, [(n+2^{m-2}) mod N, (n+2^{m-1}) mod N]，此处不是简单地将 Chord 的 Finger 表取前 m-1 项，由于将环一分为二，其中涉及到节点(n+N/2) mod N(即 Chord 环中当前节点径对的节点)的归属问题，此处为了处理的简便，将其直接划到顺时针 Finger 表内第 lb$N$-1 项的所属区间内，原来的开区间变为闭区间。

逆时针 Finger(Ccw_Finger Table)：是指存储在每个节点之上的逆时针路由表。由于是逆时针的，它在[(n-1) mod N, (n-2^{m-1}) mod N)范围内，也存储 m-1 条记录，用于存储逆时针查找的后续节点信息，划分的区间为[(n-1) mod N, (n-2) mod N), [(n-2) mod N, (n-4) mod N), …, [(n-2^{m-2}) mod N, (n-2^{m-1}) mod N]，其后序节点也描述逆时针方向的后续。需要通过收集环内部分节点的部分 Finger 表信息和本节点在 Chord 环 Finger 表中第 m 项来生成。

3.2 加入邻居表的 TB_Chord 路由表

在 TB_Chord 路由算法中，每个节点的路由表由三部分组成。第一部分是保存在整个地址空间的前驱活动节点的地址和保存整个地址空间的后继活动节点的地址，分别用 predecessor()和 suc-cessor()表示。第二部分由 Finger 表组成，包括顺时针 Finger 表和逆时针 Finger 表。第三部分是 Neighbor 表，利用已有技术将物理网络上相邻近的节点组织成簇，从而找到该节点的邻居信息。

3.3 超级节点

为减少通信量，将同一簇内近期查询结果在超级节点中进行缓存。簇内的节点一旦本次查询结束就将查询结果发送给本簇的超级节点，以便将结果加入缓存信息表。可以用先进先出(FIFO)、最近最少使用(LRU)等策略对缓存信息表进行管理。为了防止单超级节点成为本簇的瓶颈这一现象，采用文献[6]中的方法，选择性能较好的多个节点作为超级节点，每个超级节点承担相同的责任，当节点查询缓存信息表时，随机选出一个超级节点进行查询。

3.4 TB_Chord 路由算法

TB_Chord 路由发现算法描述如下：

(1)当某节点 A 需要发起一次路由查询(或接收到经转发的路由消息)时，它首先查询键值 K，如果查询键值 K∈[(predecessor(node A), node A)]，则路由发现过程终止，返回找到或者没有找到消息，否则进入步骤(2)。

(2)查询节点 A 所在簇的超级节点的缓存信息表，若找到则返回结果，否则进入步骤(3)。

(3)在 A 节点的顺时针 Finger 表、逆时针 Finger 表、Neighbor 表中，找到一个节点值小于键值 K 且与键值最接近的节点 B，重新以 B 作为发起节点，转入步骤(1)。如果没有找到，进入步骤(4)。

(4)在顺时针 Finger 表、逆时针 Finger 表、Neighbor 表中，找到一个节点值最大的节点 C，以 C 作为新的发起节点，转入步骤(1)，如此过程，直到找到目标节点为止。

在路由的过程中用到了邻居表的信息，这大大减小了路由的延迟；同时由于在路由表中增加了邻居表，而其邻居节点 ID 值是一随机的值，相当于小世界模型中随机增加了几条缓存长链，在路由算法的设计中，每次都是尽可能找最接近目标的节点进行路由，这样得到的 TB_Chord 系统模型与 Chord 系统模型相比，在覆盖网络上的路由跳数也有了很大的减少。

TB_Chord 系统的路由算法中节点的随时加入、退出，以及定期更新路由表中的各项路由信息和 Chord 中类似，在此不详述。

4 仿真实验

4.1 实验设置

本文使用 BRITE 作为拓扑产生器生成二层拓扑结构。BRITE 有 4 种类型的拓扑参数，这里选择按照 bottom-up 类型生成。按照 bottom-up 类型参数生成的拓扑有 2 层，分别是 Router-level 和节点，并且每一层都有 2 种生成模式。实验中 Router-level 拓扑使用 Waxman 模式，router 节点按照 heavy-tailed 的方法被分配到自治系统中。并用先进先出策略管理超级节点的缓存表，将簇的大小设为 100。

4.2 实验结果及分析

在实验中，从物理跳数、延迟这两方面对 TB_Chord 和 Chord 进行比较，并就超级节点数目对算法的性能产生的影响进行实验。

在比较物理跳数、延迟时将超级节点的数目设为 3，延迟的单位为 ms。如图 2 所示，在相同节点数目下，TB_Chord 的平均物理跳数少于 Chord。如图 3 所示，本算法在访问延迟方面的性能较 Chord 也有明显的提高。这是因为 TB_Chord 在选择下一跳时是从顺时针 Finger 表、逆时针 Finger 表、Neighbor 表中找一个节点值小于键值 K 且与键值最接近的节点。

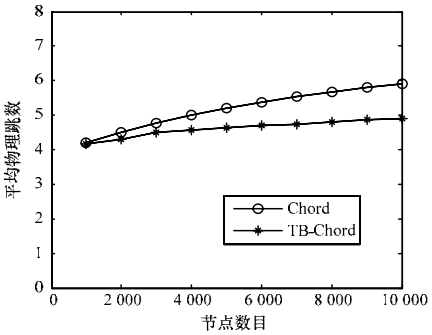

图 2 物理跳数比较

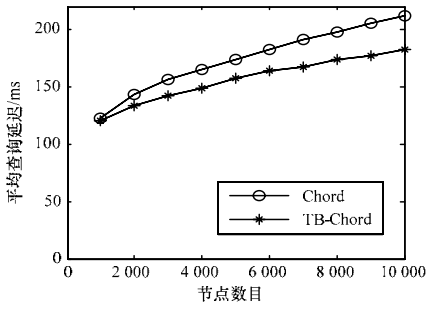

图 3 延迟比较

设节点数目为 4 096，图 4 显示了当超级节点数目 k 取不同数值时物理跳数和延迟的分布情况，可以看出，随着 k 的增大，这两方面的性能都随之提高。另一方面，超级节点越多，需要的缓存越大，并且当节点查询某关键字时，由于以随机方式查询某一超级节点的缓存信息表，将消耗簇内带宽。

(下转第 121 页)