第 36 卷 第 7 期 计 算 机 工 程 2010 年 4 月
Vol.36 No.7 Computer Engineering April 2010
·网络与通信· 文章编号：1000—3428(2010)07—0112—03 文献标识码：A 中图分类号：TP393

基于 Chord 协议的混合 P2P 模型

曾晓云
(广西财经学院计算机与信息管理系，南宁 530003)

摘 要：在结构化点对点(P2P)模型中，节点异构性会引起系统的不稳定。针对该问题，结合混合 P2P 模型的优点，构造一个基于 Chord 协议的混合 P2P 模型，将节点按处理能力分为超节点和普通节点，多个超节点被组织到同一个群组中，由超节点管理普通节点以提高系统稳定性。该模型采用基于拓扑感知的搜索算法，能较好地解决分布式哈希表(DHT)技术的路由绕路问题。实验证明，该模型在一定程度上降低查询延时，可提高查询效率。

关键词：点对点；结构化 P2P 模型；基于拓扑感知的搜索算法

Hybrid P2P Model Based on Chord Protocol

ZENG Xiao-yun
(Department of Computer and Information Management, Guangxi University of Finance and Economics, Nanning 530003)

【Abstract】Aiming at the system instability of structured P2P model which is caused by the heterogeneity of nodes in model, this paper proposes a Hybrid Structure P2P network based on the Chord(HSChord). The model is combined with the advantages of hybrid P2P model. According to the nodes' processing power, it divides nodes into super-nodes and ordinary-nodes. In this model, some super-nodes are organized into the same group, and manage the ordinary-nodes to improve system stability. The model proposes search algorithm which is based on topology-aware. The algorithm can solve the routing detour which is caused by DHT technical inquiries better. Experimental results show that the search algorithm can reduce the query delay and improve the efficiency of query.

【Key words】P2P; structured P2P model; search algorithm based on topology-aware

1 概述

P2P 网络拓扑模型和搜索技术是网络应用研究的热点问题。其中，基于分布式哈希表(DHT)技术的结构化网络具有无中心控制、可扩展性强及高效的查询等特性。但是 DHT 技术会破坏节点的物理拓扑位置信息，导致节点间产生“路由绕路”[1]问题。其次，DHT 技术没有考虑节点的异构性，节点的频繁加入、退出会造成网络路由波动。此外，性能较差的节点直接加入系统会成为潜在的瓶颈。而混合结构的 P2P 系统能较好地考虑节点的异构性，使不同性能的设备承担不同的任务。

本文在 Chord 协议的基础上，结合混合 P2P 模型的优点，构造基于 Chord 协议的混合结构 P2P 网络模型(Hybrid Structure P2P network based on the Chord, HSChord)并设计相应的搜索算法解决上述问题。

2 HSChord的体系结构和基本原理

2.1 HSChord模型的体系结构

在 HSChord 模型中，每个节点都有唯一的 ID 和 IP 地址，多个节点组成一个群组，设 G_i 表示群组的 ID，S_{i_j} 表示某个群组 G_i 的超节点。

HSChord 模型的体系结构分为上下 2 层。Client 层位于模型的下层，以群组的形式组织，由物理距离较近的节点组成。每个群组有一个或多个超节点；能力较弱的普通节点是超节点的叶子节点。

Group 层位于模型的上层，是基于 Chord 协议组织的一个结构化 P2P 网络覆盖层。Chord 环上的每个节点是包含多个节点的群组(也称为虚拟节点)，群组里的每个超节点都要保存并维护一个指针表 FingerTable，用于保存 Chord 环上群组之间的路由关系。

在 HSChord 模型中，FingerTable 中的 SuccessorGroup 表示后继群组，其实就是管理这个后继群组的所有超节点的信息列表。HSChord 模型的结构如图 1 所示，群组 G_8 的超节点 S_{8_1} 和 S_{8_2} 的 FingerTable 的结构如图 2 所示。

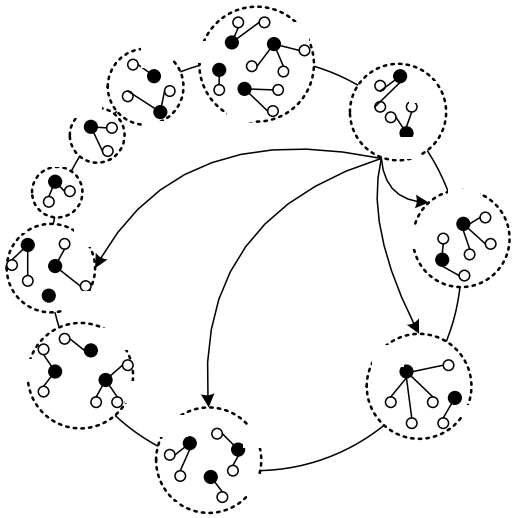

图 1 HSChord 模型结构

作者简介：曾晓云(1980－)，女，讲师、硕士研究生，主研方向：网络应用，计算机软件理论

收稿日期：2009-11-20 **E-mail：**xxxxxx2003@126.com

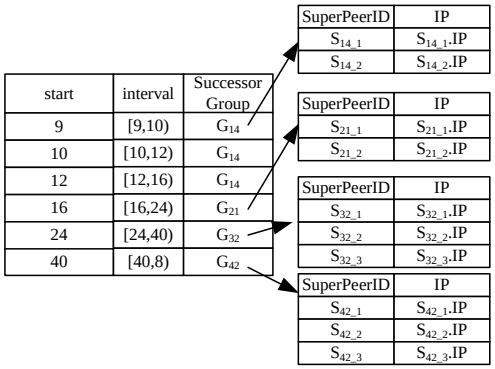

图 2　群组 G8 的超节点 S_{8_1} 和 S_{8_2} 的 FingerTable 结构

Group 层的主要作用是利用 Chord 环的特点，使用 DHT 的高效定位查询算法，完成全网范围内的节点的发现定位和路由。

2.2　拓扑感知的基本原理

为进一步降低查询延时，本文根据 PNS[2]原则，使用 Vivaldi[3]来对 FingerTable 进行改进。Vivaldi[3]是一个网络坐标系统，该坐标系统可以根据节点的坐标去预测节点之间的延迟。

本文对超节点的指针表 FingerTable 的改进如下：使用 Vivaldi[3]计算群组 n 的某个超节点 S 与区间$[n+2^i, n+2^{i+1})$中最开始的 x 个群组中各超节点的距离，找出与 S 延迟最小的群组；在 FingerTable 上加一列 NearGroup，将选出的延迟最小的群组作为 NearGroup 的第 i 项。改进后的 FingerTable 结构如图 3 所示(虚线箭头表示在区间$[n+2^i, n+2^{i+1})$中与当前超节点延迟最小的群组)。

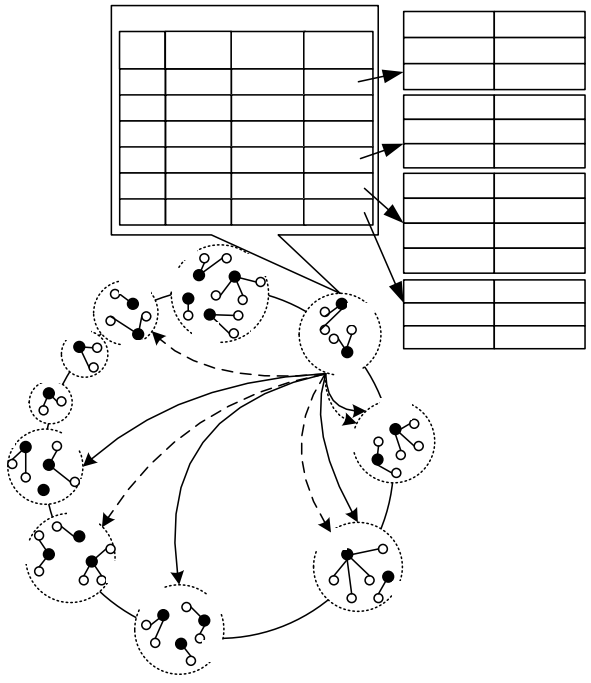

图 3　改进的 FingerTable 结构

2.3　HSChord模型的数据结构

群组中的各超节点还要保存一个超节点列表(SPTable)，记录所在群组的所有超节点的信息，以图 3 中群组 G_8 的超节点为例，它们的 SPTable 表结构如表 1 所示。为方便管理共享资源，超节点还保存一个资源索引表(ResourceTable)，结构如表 2 所示。每个超节点要用表 PredecessorGroup 来保存自己所在群组的直接前趋群组信息，结构如表 3 所示。同一群组里的 FingerTable, SPTable, ResourceTable 和 Predecessor Group 要保持一致。

表 1　群组 G_8 的超节点列表(SPTable)

SuperpeerID	IP	Ability
S_{8_1}	122.35.13.1	C_{8_1}
S_{8_2}	122.35.9.25	C_{8_2}

表 2　共享资源索引表(ResourceTable)

keyID	ResourceIP
0x3768	13.0.42.36
0x3760	13.0.73.13

表 3　群组 G8 的直接前驱信息表(PredecessorGroup)

SuperpeerID	IP
S_{1_1}	S_{1_1}. IP
S_{1_2}	S_{1_2}. IP
S_{1_3}	S_{1_3}. IP
S_{1_4}	S_{1_4}. IP

3　HSChord模型的维护

3.1　HSChord模型的生成和节点的加入

创建群组的节点被认为是群组的第一个超节点，称为群首节点，该群组的 ID 是对群首节点的 IP 地址进行哈希后得到的。新节点请求加入时，首先判断其能力是否达到超节点的要求，如果是，则进一步判断该节点与其他群组的距离大小，从而决定新节点是作为超节点加入已有群组还是作为群首节点生成新的群组；如果请求加入的节点达不到成为超节点的要求，则作为普通节点加入某个已存在的群组。设 A 是请求加入的节点，S 是与 A 距离最近的超节点，d_{AS} 表示 A 与该超节点 S 的距离，由于同一个组群内的超节点彼此距离较近，因此可以近似地认为 d_{AS} 是节点 A 与 S 所在群组的距离，d 表示设定的距离阈值；C_A 表示节点 A 的能力，C 表示设定的能力阈值；m_G 表示当前群组 G 的超节点数目，m 表示设定的群组超节点数阈值。下面将对新节点加入的 3 种情况分别进行详细描述：

(1)新节点作为超节点加入已存在的群组

如果 A 的处理能力大于设定的能力阈值 C，且 A 与某个群组的距离较近($\leqslant d$)，则 A 以超节点的身份加入与该群组。

加入过程的形式化描述如下：

1)在离 A 较近的群组(两者的距离$\leqslant d$)中选择一个与 A 距离最近的群组，转入 2)。

2)A 向该群组中的群首节点发送加入请求，如果当前群组的超节点数目 m_G 未达到 m，则转入 3)。否则转入 5)。

3)允许 A 作为超节点加入，群首节点将 A 的信息加入自己的 SPTable；然后将自己的 FingerTable, PredeccessorGroup, ResourceTable 和 SPTable 复制给 A，并通知本群组的其他超节点修改各自的 SPTable。

4)A 通知直接后继群组的所有超节点，将 A 的信息添加入它们的 PredeccessorGroup。加入过程结束。

5)A 重新选择下一个与之距离最近的群组($\leqslant d$)，若该群组存在，则转入 2)。否则 A 就作为群首节点创建新的群组加入 Chord 环，加入过程结束。

(2)新节点加入并形成新的群组

如果一个节点 A 作为群组首节点创建新的群组加入 Chord 环，则它需要借助任意一个已知的超节点 S 进行引导。因为群组是 Chord 环上的虚拟节点，群首节点可以代表一个群组，而新加入的群组其实是一个节点，所以新节点 A 作为群组加入 Chord 环的过程类似于传统 Chord 环中的新节点加入。

加入过程的形式化描述如下:

1)A 初始化自身 FingerTable 中 start 域。

2)向引导节点 S 发送定位消息，定位 finger[1].start 的后继群组，这个后继群组也同样是新加入节点 A 的后继群组。

3)A 根据其直接后继群组的超节点的 PredecessorGroup 更新自身的 PredecessorGroup，然后用 A 的值更新直接后继群组中所有超节点的 PredecessorGroup。

4)新加入的群首节点 A 向后继群组发送定位消息，初始化 FingerTable 中的其他 SuccessorGroup 域。

5)新加入的群首节点 A 调用定位过程定位 A 的前趋群组，并向前趋群组的群首节点发送更新 FingerTable 的消息。

6)这些群首节点对 FingerTable 中的 finger[i].Successor Group 进行更新。

7)A 使用 Vivaldi[3]计算区间[A+2^i,A+2^{i+1})中与 A 延迟最小的群组，并更新自身 FingerTable 的 NearGroup 列。

8)其他群首节点通知本群组的超节点更新 FingerTable。

9)把所有后继群组是 A 的共享资源索引转移到 A 上。

(3)新节点作为普通节点加入

新节点 A 作为一个普通节点加入网络的具体操作如下:

1)A 从已存在的群组中选择一个与之距离最近的群组，转入 2)。

2)A 向该群组中的群首节点发送加入请求，群首节点从自己的 SPTable 中选另一个能力强、负担轻的超节点回复 A 的请求，完成加入。

3.2 节点退出

节点退出分为普通节点退出和超节点退出(包括群首节点和一般超节点)。普通节点退出时直接离开网络即可，不需要向超节点汇报。超节点的退出则相对比较复杂，其退出过程描述如下:

(1)群首节点的退出

群首节点负责群组中各超节点的 FingerTable 的更新。为避免群首节点意外退出网络造成的麻烦，群组中的各超节点(包括群首节点)周期性地向其他超节点发送心跳和自身处理能力的信息，表示自己在线。并且群首节点周期性地将其他超节点的处理能力进行排序，找出处理能力最强的超节点作为自己的候补。

群首节点主动要求退出网络的过程如下:

1)群首节点向同群组的其他超节点发送退出信息。

2)这些超节点收到后修改自己的 SPTable,将原群首节点的信息从 SPTable 中删去，并将事先选好的候补群首节点作为新的群首节点。

如果各超节点在一段时间内没有收到群首节点发送来的心跳信息，则认为群首节点掉线，与该群首节点处于同一群组的超节点修改自己的 SPTable，将原群首节点的信息从 SPTable 中删去，并将事先选好的候补群首节点作为新的群首节点。

(2)超节点的退出

超节点 S 主动要求退出网络的过程如下:

(1)S 向同群组的其他超节点发送退出信息;

(2)这些超节点收到后修改自己的 SPTable，将 S 的信息删去。

如果其他超节点在一段时间内没有收到 S 发送的心跳信息，则认为 S 掉线，与 S 处于同一群组的其他超节点修改自己的 SPTable，将 S 的信息删去。

3.3 HSChord模型中基于拓扑感知的搜索算法

HSChord 模型使用拓扑感知的方法修改 FingerTable，增加了 NearGroup，使查询的每一步都向着[$n+2^i$, $n+2^{i+1}$)中与当前超节点延迟最小的群组进行，相应的搜索算法描述如下:

(1)某个节点 A 向管理自己的超节点 S 发送查询 R 的请求。

(2)S 对 R 的关键字进行哈希，得到 KeyID。

(3)S 判断 KeyID 是否在自身群组标志和直接后继群组标志之间，如果是，则当前群组的后继群组是目标群组，转(4);否则，转(5)。

(4)S 将搜索请求发送给直接后继群组的某个超节点，该超节点收到后在自己的 ResourceTable 上搜索，并返回搜索结果。

(5)S 查找自己的 FingerTable，找到 NearGroup 上群组标志最大但不超过 KeyID 的最后一个群组 G，将搜索请求发送至 G 的某个超节点 S'。

(6)S'重复(3)~(5)，最终返回结果。

4 性能分析

由于群组内部有多个超节点，使得群组退出网络的概率非常低。假设一个超节点意外失效的概率是 p，群组内超节点的数目是 n，则通过计算概率可以得到整个群组中所有的超节点都失效的概率为 p^n(即整个群组瘫痪的概率)。但是每个群组中只要有一个超节点不失效，这个群组就不会瘫痪，所以某个群组不瘫痪的概率为 1-p^n。对于较大的 n，p^n 非常小，即使超节点失效的概率为 50%，拥有多个超节点的群组瘫痪的概率仍然非常小。因此，若群组中的超节点数大于或等于 5，就基本上可以保证该群组正常运行。

本文使用 OMNET++3.2 作为仿真工具。主要从平均路由跳数和搜索延迟延伸率 2 个方面评估 HSChord 的性能。模拟过程中每次选择总节点数的 5%作为超节点。

图 4 给出 Chord 系统与 HSChord 系统的平均消息路由跳数和系统节点数的关系。理论上 Chord 路由平均跳数为 1/2log(N)，而 HSChord 体系的平均路由跳数可以减少到 1/2log(I)，其中，I 是群组的数量。实验结果和从理论上分析的结果基本一致。

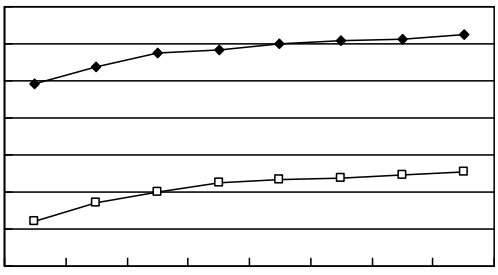

图 4 HSChord 与 Chord 平均路由跳数的比较

延迟延伸率 latency stretch[4]是查询消息在覆盖网络中所经过的距离和在底层网络中实际距离的比值。

从图 5 可看出，随着网络规模的增大，没有使用基于拓扑感知的 HSChord 网络的延迟延伸率增长很快，而使用了基于拓扑感知的 HSChord 网络，其搜索延迟延伸率较小。这说明使用了拓扑感知查询方法的 HSChord 模型考虑了群组底层的物理地址，提高了网络路由的性能。

(下转第 118 页)