基于分层象限空间的 P2P 超级节点拓扑构造

冯劲潇 陈贵海 谢俊元
(南京大学计算机软件新技术国家重点实验室 南京 210093)

摘 要 现有的 P2P 超级节点拓扑分为非结构化超级节点拓扑和结构化超级节点拓扑,前者支持模糊查询,鲁棒性好,但路由效率低,查询结果具有不确定性,后者支持数据定位,路由效率高,负载均衡性好,但不支持模糊查询,也不适应高度动态环境.文中将两种超级节点拓扑结构的优点结合起来,提出一种基于分层象限空间的新型超级节点拓扑 Quad.它在路由方面同时支持数据定位和模糊查询,并解决了结构化超级节点拓扑对动态性支持差的问题.分析和模拟结果表明:和现有的超级节点拓扑相比,Quad 数据定位在常数度数下达到 $O(\log N)$ 定位性能,并在路由效率和路由状态数上获得更好的权衡.在动态环境中,Quad 能更有效地处理超级节点失效;在性能上,它具有更少的拓扑构建和失效修复成本并具有良好的负载均衡性.

关键词 对等网络;超级节点拓扑;分层象限空间;负载均衡
中图法分类号 TP393 **DOI 号**: 10.3724/SP.J.1016.2010.00988

P2P Super-Peer Topology Construction Based on Hierarchical Quadrant Space

FENG Jin-Xiao CHEN Gui-Hai XIE Jun-Yuan
(*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

Abstract Current P2P super-peer topology generation methods consist of unstructured and structured super-peer topology. The former is robust and supports proximity search, but its routing efficiency is lower and search result is uncertain. The latter supports data locating and has high routing performance and good load balance, but it does not provide proximity search and is not suitable for high dynamic environment. In this paper the authors combine the advantage of both topology structures and introduce a new kind of super-peer topology generation method which is based on hierarchical quadrant space and called Quad. Quad supports both data locating and proximity search and solves the issues that structured super-peer topology can not support the dynamic environment well. Analysis and simulation show that compared with current super-peer topology, Quad offers $O(\log N)$ routing performance with constant degree and attains a better tradeoff between routing efficiency and routing state size in data locating. Quad can handle the super-peer failure more effective under high dynamic environment. Meanwhile Quad keeps less topology construction and failure repair cost and has good performance in load balance.

Keywords peer-to-peer network; super-peer topology; hierarchical quadrant space; load balance

1 引 言

经过近几年的发展,P2P 网络已成为互联网最重要的应用之一,从文件下载应用 eMule、BitTorrent 到语音通信软件 Skype,再到流媒体网络电视 PPLive、PPStream 等,每一种新应用出现都给互联网带来巨大的冲击,P2P 网络正成为一种流行的基

收稿日期:2009-03-09;最终修改稿收到日期:2010-01-12.本课题得到国家"九七三"重点基础研究发展规划项目基金(2006CB303000)、国家自然科学基金(60721002)、国家自然科学杰出青年基金(60825205)资助.**冯劲潇**,男,1976 年生,博士研究生,研究方向为 P2P 网络、信息安全.E-mail: fjx_mailbox@126.com. **陈贵海**,男,1963 年生,教授,博士生导师,研究领域为并行计算和无线网络. **谢俊元**,男,1961 年生,教授,博士生导师,研究领域为人工智能、信息安全.

础架构,对互联网应用和分布式研究产生深远影响.

P2P 网络是在底层 TCP/IP 网络基础上构成的应用层覆盖网络. P2P 拓扑结构目前主要包括非结构化和结构化两种类型. 在非结构化拓扑中,Peer 节点间无严格组织,资源存放在节点本地,路由传递具有随意性,其优点是系统鲁棒性好,节点失效影响小,缺点在于查找时需要通过泛洪或随机游走等盲目查找方式进行,消息流量大或查询效率较低. 为改进非结构化系统的路由效率,研究人员提出基于分布式 Hash 表(DHT)的结构化系统,这类系统采用一致性 Hash 函数将节点标识(如 IP 地址)和资源标识(如文件名)分别映射为节点 ID 标识和资源 ID 标识,资源信息统一存放在节点 ID 标识与该资源 ID 标识最相近的节点上(资源信息可以是资源本身内容,也可以是资源位置信息,相关节点为资源信息建立索引表,所以本文用资源索引来简化表示节点存放的资源信息). 与非结构化拓扑相比,DHT 机制保证系统能在 $O(\log N)$ 上限内查找到指定的资源索引,路由效率有了很大改善. 但结构化拓扑也带来两个新的问题:一是不适应高度动态环境,原因是结构化系统具有严格的拓扑,资源索引要发布到与节点 ID 标识最相近的节点上,一旦节点失效,储存在该节点的资源索引需要转移或重新发布. 另外,路由表中往往包含 $O(\log N)$ 个邻居节点,这些邻居节点关系紧密,节点一旦失效会产生很高的拓扑修复费用;另一个问题是结构化系统支持数据定位但不支持模糊查询,这是因为不同的资源标识产生不同的 ID 标识. 鉴于上述不足,研究人员利用超级节点结构对非结构化和结构化拓扑进行改进,称之为非结构化超级节点拓扑和结构化超级节点拓扑. 目前的超级节点结构虽然在这两种拓扑中有效利用了节点异构性,但仍然没有解决非结构化拓扑路由效率不高以及结构化拓扑对动态性和模糊查询支持差等问题. 事实上,这两类结构具有互补性,非结构化超级节点拓扑支持模糊查询、对动态环境支持好,而结构化超级节点拓扑路由效率高,良好的拓扑结构需要同时具备两种结构的优点.

本文提出一种新颖的分层象限空间(Hierarchical Quadrant Space)结构,并在该结构上建立了一种新型超级节点拓扑,称为 Quad,它将非结构化和结构化两种超级节点拓扑优点结合起来,在路由方面同时支持模糊查询和数据定位,并解决了结构化超级节点拓扑对动态性支持差的问题. 分析和模拟结果表明:与现有超级节点拓扑相比,Quad 数据定位在空间(常数度数)和时间($O(\log N)$定位性能)上获得更好权衡;在动态适应性方面,Quad 能更好地降低超级节点失效对拓扑产生的影响. 在性能上,Quad 具有更小的拓扑构造和失效修复成本并具有良好的负载均衡性.

第 2 节介绍 P2P 拓扑构造相关工作;第 3 节是 Quad 网络模型定义;第 4 节描述 Quad 对模糊查询和数据定位两种路由方式支持;第 5 节介绍 Quad 拓扑自组织、负载均衡和容错机制;第 6 节分析比较 Quad 和结构化超级节点拓扑在查找路径长度、路由状态数、失效修复成本以及负载均衡方面的差别,提出 Quad 路由改进思路;第 7 节通过模拟程序比较 Quad 和非结构化超级节点网络在拓扑构造、失效修复等方面的成本差异;第 8 节给出结束语.

2 相关工作

P2P 拓扑结构按分布式特征分为 4 类:集中索引式拓扑、完全分布式非结构化拓扑、完全分布式结构化拓扑和超级节点拓扑,相关研究工作如下:

(1)集中索引式拓扑和完全分布式非结构化拓扑. 最早的 P2P 应用 Napster① 采用集中索引式拓扑,这种拓扑结构将所有节点的资源索引存放在中心服务器,查询通过中心服务器进行,该结构虽然简单,但是具有单点失效和扩展性差等缺点. 在 Napster 出现不久,完全分布式非结构化拓扑就出现了,典型应用包括 Gnutella0.4 版② 和 Freenet[1] 等. 在这类结构中,对等节点将资源索引存放在本地,查找主要采用洪泛和随机游走两种方式,但前者产生很大的网络流量,后者查找路径长,效率偏低. 针对完全分布式非结构化拓扑的改进主要是提高查询效率,如基于 Bloom Filter 技术的概率搜索小组算法[2] 等. 但总体来说,查询仍然是随机的,查询结果也是不确定的.

(2)完全分布式结构化拓扑. 在基于 DHT 的完全分布式拓扑中,资源索引的存放具有严格的限制,节点和资源标识都通过一致性 Hash 函数映射为数字 ID 标识,资源索引存放在节点 ID 标识与该资源 ID 标识最相近的节点上,典型的 DHT 系统包括 Chord[3]、CAN[4]、Pastry[5]、Tapestry[6]、Kademlia[7] 等. 在完全分布式结构化系统中,查询在 $O(\log N)$

① Napster. http://www.napster.com
② The Gnutella Protocol Specification v0.4. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

上限内可以完成，可扩展性较好，不足之处在于 DHT 机制只支持精确查询，不适应动态性很强的 P2P 网络，限制了这种结构在文件共享等领域的应用. 一类常数度数的结构化系统，如 Viceroy[8]、Koorde[9]、Cycloid[10] 等对结构化系统进行了改进. 在常数度数结构中，路由状态数是固定的，不随网络节点规模增大而变化，因而可以减少拓扑维护成本，但这些系统并不能解决结构化拓扑对动态性和模糊查询支持差的问题. 文献[11]提出一种混合式结构，即在非结构化拓扑上建立查找覆盖网，在结构化拓扑上建立索引覆盖网，该结构采用单层结构，不具备超级节点拓扑性能高、稳定性好等优点，在动态性强的 P2P 环境下也面临结构化索引维护成本过高的问题.

(3) 超级节点拓扑. 超级节点拓扑按超级节点层是否支持 DHT 路由分为非结构化超级节点拓扑和结构化超级节点拓扑. 前者包括 Gnutella0.6 版①、Fasttrack② 协议等，系统分为两层，上层由能力较强的节点组成，称为超级节点，下层是普通节点，普通节点不参与系统的路由，查询通过超级节点进行. 由于减少了参与路由的节点数，这类超级节点结构比传统的非结构化拓扑具有更好的查询性能，但路由长度和查询结果仍具有不确定性. 后一类超级节点结构在超级节点层支持 DHT 机制，叶节点间可以采用 DHT，如文献[12]，也可不采用 DHT，而是直接连到超级节点，叶节点通过超级节点发布和查询内容，如 IS-P2P[13] 模型. 在结构化超级节点拓扑中，由超级节点构成的路由层比由完全分布式结构化拓扑中所有节点构成的路由层更加稳定. 同时，由于参与路由的超级节点数量只占全部节点的一部分，因而查找路径长度更短. 不足之处在于超级节点一旦失效带来的影响更大，不仅需要修复拓扑，结构化资源索引也需要重新发布或进行转移，该结构也不支持模糊查询.

与现有超级节点拓扑不同，本文提出一种新颖的基于分层象限空间的 P2P 超级节点拓扑 Quad，它将非结构化和结构化两种超级节点拓扑优点结合起来，其具体特征如下：

(1) 将分层象限空间运用于超级节点拓扑构造. 象限空间的每个位置都有唯一的数字标识，称为位置标识，超级节点在拓扑中的位置不再由节点 ID 标识决定，而是由位置标识确定. 这样，Quad 既可以实现在超级节点层按资源 ID 进行定位，又可以避免超级节点失效给拓扑带来很大影响，因为新的替换超级节点与失效超级节点具有相同的位置标识，拓扑易于修复. 另外分层象限空间使超级节点拓扑具有常数度数特点，维护成本低.

(2) 支持模糊查询和数据定位两种路由方式. 分层象限空间类似于树形结构，支持广度优先(BFS)和深度优先(DFS)遍历，另外分层象限空间本身又是一个数值空间，所以 Quad 既可以支持泛洪和随机游走等模糊查询方法，又可以按资源 ID 进行数据定位.

(3) 拓扑具有良好的负载均衡性和容错机制. Quad 具有自适应的调整、分裂与合并等负载均衡机制，超级节点可以向不同层次的多个邻居超级节点转移负载. 在容错性方面，Quad 具有周期性检测和冗余路径两种方法，在动态性强的环境中具有容错能力.

3　Quad 网络模型定义

定义 1.　象限空间是一种二维平面空间，以原点为基准，横轴和纵轴将平面划分成 4 个象限，从原点的右轴开始，象限空间按逆时针方向分为一、二、三和四象限.

定义 2.　分层象限空间是象限空间的扩展，象限空间的每个象限可继续划分为 4 个子象限，逐层划分，就形成分层象限空间.

分层象限空间思想借鉴了树形结构的层次概念，但是两者并不相同. 与树形结构相比，分层象限空间在拓扑构造上具有两点优势：首先，象限空间比树形结构具有更直观的方向标识能力，无论是按序构造还是随机构造每个象限位置方向都非常清晰；其次，在树形结构中，上下层节点之间是一对多关系，标准的树形结构中节点没有同层邻居. 而在分层象限空间中，上下层节点之间包含一对多和多对多两种关系，节点有同层邻居，因而分层象限空间比树形结构更易于实现负载均衡和容错.

Quad 系统中超级节点按分层象限空间进行组织，如图 1 所示. 象限中心点 A_0 有 A_1、A_{12}、A_2、A_{23}、A_3、A_{34}、A_4 和 A_{41} 8 个邻居，每个邻居节点包含一个方向，按逆时针顺序分别表示为二进制形式的(000，001，010，011，100，101，110，111)或八进制形式的 0～7. 第 1 层包含 A_0、A_1、A_2、A_3、A_4 5 个超级节点，

① Gnutella protocol spec. v. 0. 6. http://rfc-gnutella.sourceforge.net/src/rfc－0_6-draft.html

② Fasttrack. http://en.wikipedia.org/wiki/FastTrack

它们互为同层邻居，A_0 包含 A_{12}、A_{23}、A_{34}、A_{41} 4 个位于第 2 层的下层邻居，第一象限的 A_1 包含 B_1、B_2、B_3、B_4、A_{12} 5 个位于第 2 层的下层邻居，其余 3 个象限类似. 在 Quad 象限空间中，每个位置的标识值是唯一的，由 A_0（不包含 A_0）到该节点路径包含的节点方向构成，如 A_{12} 位置标识为 001，B_1 位置标识为 001000，A_0 可以设为一个特定的标识并在系统中作为一个特例进行处理.

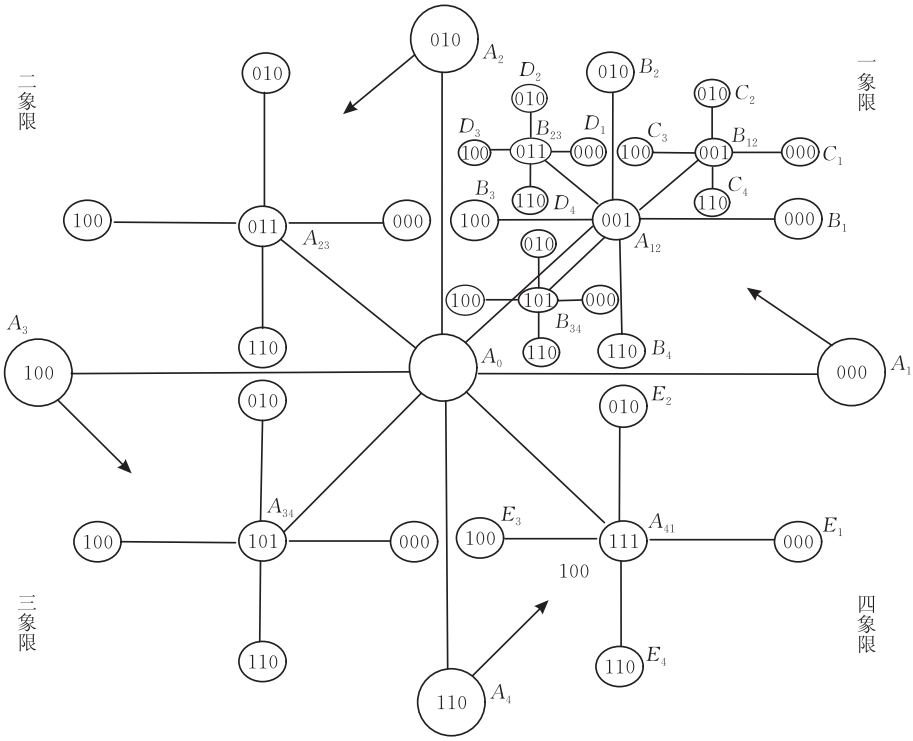

图 1 Quad 超级节点结构

定义 3. 超级节点定义为七元组 $S=(I,PI,C,D,R,L,Y)$，I 是超级节点 ID 标识，PI 为超级节点位置标识，当超级节点离开时，该标识分配给处于相同位置的新超级节点使用. C 表示节点容量，D 为节点当前负载，R 为节点路由表，L 表示该节点所处层次，从上层到下层层次值依次增加. Y 为节点类型，超级节点分为两种类型，象限横轴和纵轴的交点称为中心超级节点 *CSP*(Center Super-Peer)，横轴和纵轴的 4 个端点构成 4 个端超级节点 *BSP* (Boundary Super-Peer).

定义 4. 叶节点定义为四元组 $F=(I,C,Y,S)$，I 为叶节点 ID 标识，C 表示节点容量，Y 为叶节点类型，叶节点分为两类：一类是普通叶节点，另一类是候选超级节点. 每个叶节点通过一个超级节点加入到网络中，超级节点按照能力和在线时间等综合因素动态从普通叶节点中选拔最优秀的叶节点作为候选超级节点，候选超级节点对父超级节点路由表和资源索引进行备份，并定期和超级节点交换信息，以便在超级节点失效时能接替其位置. S 表示超级节点，是叶节点的网络接入点.

定义 5. 超级节点 S 的路由表定义为二元组 $R=(NR,QR)$，NR 是邻居路由表，记录超级节点与同层、下层和上层邻居超级节点之间的路由关系. QR 是象限路由表，支持顶层象限不相同的超级节点之间定位.

定义 6. 超级节点 S 的邻居路由表定义为三元组 $NR=(SNR,CNR,PNR)$，SNR 是同层超级节点邻居路由表，其表项由同一层次的 *CSP* 和 *BSP* 构成；CNR 是下层超级节点邻居路由表，表项由下层中心超级节点（*CCSP*）和下层端超级节点（*CBSP*）构成；PNR 是上层超级节点邻居路由表，表项包含上层中心超级节点（*PCSP*）和上层端超级节点（*PBSP*）两种类型.

以图 1 为例，中心超级节点 A_{12} 的邻居路由表为 $SNR(B_1,B_2,B_3,B_4)$，$CNR(B_{12},B_{23},B_{34})$，$PNR(A_0,A_1)$. 端超级节点 B_1 的路由表为 $SNR(A_{12},B_2,B_3,B_4)$，$CNR(C_1,C_2,C_3,C_4,B_{12})$，$PNR(A_1)$. 在 Quad 中，超级节点邻居路由表中节点数上限为 10，不随网络规模增大而同步变化.

NR 中用索引 0～9 标识 10 项邻居节点信息，

如表 1 所示. *CSP* 用索引(0,2,4,6)标识同层的 *BSP*, *BSP* 用索引(0,2,4,6)标识 *CSP* 和其它 3 个 *BSP*,其中该 *BSP* 节点自身方向对应的索引用来存储 *CSP*; *CSP* 的 *CNR* 为 4 个象限的子中心超级节点(*CCSP*),索引为(1,3,5,7),*BSP* 的 *CNR* 为对应象限的 4 个子端超级节点(*CBSP*)和子中心超级节点,用索引(1,3,5,7,8)表示;*CSP* 的 *PNR* 包含 *PCSP* 和 *PBSP*,用路由索引 8 和 9 表示, *BSP* 的 *PNR* 只包含 *PBSP*,用路由索引 9 表示. 容易看出, *CSP* 的 4 个同层 *BSP* 和 4 个下层 *CCSP* 的索引号与这些节点的方向是一致的,所以超级节点的位置标识(*PI*)由距离其最近的 *CSP* 位置标识加节点自身所包含的方向构成.

表 1　邻居路由表

超级节点	*SNR*	*CNR*	*PNR*
CSP	0,2,4,6	1,3,5,7	8,9
BSP	0,2,4,6	1,3,5,7,8	9

定义 7.　超级节点 S 的象限路由表定义为数组 $QR=QRA[i]$,$0\leq i<3$,其中每一个数组项表示为三元组 $QRA_i=(Q_i,S_1,S_2)$,满足:(1) $shl(PI_s,Q_i)=0$; (2) $(L_{s_1}<L_{s_2})\wedge(L_{s_2}\leq L_s)$. 其中,$0\leq Q_i<4$,标识 4 个顶层象限. *shl* 判断超级节点 S 的位置标识与 Q_i 匹配象限的长度. 方法是取 S 位置标识中第 1 组 3 个二进制位的前二位值和 Q_i 进行比较,0 表示不相等. 条件(1)表明 Q_i 是与 S 顶层象限不相同的其它 3 个象限,条件(2)表示每个象限路由表表项最多为两个超级节点(可以为空),这两个超级节点处于不同层并且其层次都不超过 S 的层次.

因为顶层象限只有 4 个,所以超级节点象限路由表(*QR*)表项只需记录属于其它 3 个顶层象限(包括顶层象限的下层象限)内的超级节点即可,由于每个顶层象限内的表项最多包含两个节点,所以象限路由表最多包含 6 项. Quad 对于象限路由表的节点要求并不严格,因为只需要知道特定象限的一个节点就可以定位到该象限,取不同层并且层次与本节点相近的两个节点是为了降低超级节点之间路由项的重复率. 另外,将每个象限的路由表项数设为 2 是为了增加系统的容错性. Quad 路由表的最优值设计可以根据具体应用在路由性能和维护代价之间进行权衡,增加路由表项会缩短路由长度,但会增加系统维护开销.

定义 8.　Quad 网络模型定义为四元组 $M=(T,V,E,X)$,如图 2 所示. T 表示拓扑逻辑结构, Quad 采用两层拓扑结构,从逻辑上分为超级节点层和叶节点层. V 为节点集,由超级节点和叶节点组成,超级节点是拓扑的核心,担负着系统的路由任务,并负责为叶节点建立和发布索引,叶节点通过超级节点进行查询. E 为拓扑路由关系,由所有超级节点路由表构成的连通关系组成. X 为资源索引,包含非结构化和结构化索引两个部分.

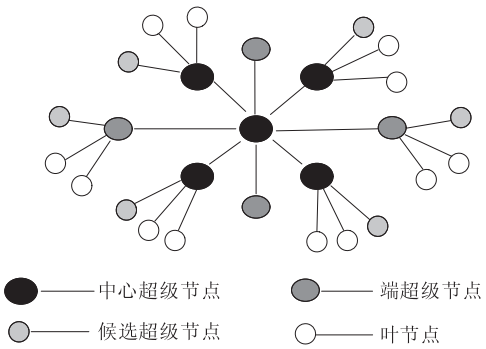

图 2　Quad 网络模型

定理 1.　在完全 Quad 系统中,超级节点层的最大层次为 $\log_4\left(\frac{3N}{5}+1\right)$.

证明.　在完全 Quad 系统中,超级节点在各象限均匀分布并且各层象限空间位置均已被分配. 在这种情况下,超级节点层的第 1 层包括 1 个中心超级节点和 4 个端超级节点,即 5 个超级节点,第 2 层每个子象限包括 5 个超级节点,依次类推,每层超级节点数可以表示为 $4^{i-1}\times 5$, i 为超级节点所在层次. 设超级节点总数为 N,超级节点最大层次为 ML,则 N 用公式表示为

$$\begin{aligned}N&=\sum_{i=1}^{ML}4^{i-1}\times 5\\&=5+4\times 5+4^2\times 5+4^3\times 5+\cdots+4^{ML-1}\times 5\\&=5\times\frac{1-4^{ML}}{1-4}=\frac{5(4^{ML}-1)}{3}\end{aligned}\tag{1}$$

对式(1)进行移项变换,得

$$4^{ML}=\frac{3N}{5}+1\geq ML=\log_4\left(\frac{3N}{5}+1\right)\tag{2}$$

证毕.

4　Quad 路由

Quad 路由包括非结构化和结构化两种方式. 在非结构化路由中,Quad 支持泛洪和随机游走等模糊查找方法. 对于泛洪方式来说,由于它采用分层结构,可以消除 Gnutella 协议中存在的消息重复问

题.在随机游走查找方法上,Quad 可以采用 Bloom Filter[14]方式建立压缩索引并在一定层次内共享,能够减少消息传递到底层时的回溯成本并缩短查找路径长度.另外,由于 Quad 同时支持超级节点结构和结构化特性,它可以将具有相似主题资源的节点更好地聚集在一起.

在结构化路由方面,Quad 将资源标识(如文件名称)通过一致性 Hash 函数(如 SHA-1 等散列算法)映射为资源 ID 标识.它和节点的位置标识都是数字 ID 标识,资源通过 ID 标识发布到分层象限空间内的超级节点,其中节点位置标识与资源 ID 标识匹配象限最长的超级节点存放该资源索引.在查找时超级节点通过查询路由表来逐步实现定位.当超级节点(设为 S)接到叶节点资源发布或查询请求后,首先判断自身的位置标识与资源 ID 标识(设为 K)是否属于同一顶层象限,如果不是,则通过象限路由表将请求传递给与 K 顶层象限相同且象限序列匹配最长的超级节点.如果属于同一顶层象限,则先定位到位置标识与 K 象限序列最长交集所对应的超级节点,再从该节点按照邻居路由表逐层向下定位到目标超级节点.如果 S 是图 1 中象限中心点 A_0,则直接向下层定位,在路由中属于特例.因为其它超级节点的象限路由表中均不含 A_0,所以只有 A_0 的叶节点发布或查询资源时才会出现 S 为 A_0 的情况,为节省篇幅文中不再考虑该特例.Quad 具体路由过程见算法 1,为叙述方便,这里先定义算法中用到的缩略语,如表 2 所示.

表 2 算法缩略语

符号	含义
$shl(PI_s,K)$	S 位置标识与 K 象限序列匹配长度
D_s^k	S 查找象限路由表,定位到位置标识与 K 象限序列匹配最长的超级节点
$FWD(S,n)$	同一顶层象限内,向上定位到 S 位置标识与 K 象限序列最长交集对应的 CSP,距离为 n 跳.
$GN(S,g)$	取 S 邻居路由表中 g 值索引的超级节点
G_K	将数字标识 K 按 3 个二进制位为一组进行划分后所产生的组数
G_K^l	表示 G_K 中的第 l 个分组
L_s	超级节点 S 的层次

象限序列表示数字 ID 标识(如位置标识和资源 ID 标识)中包含的有序象限组.将 ID 标识按 3 个二进制位为一组进行划分,每一组包含 1 个象限,用前两个二进制位计算象限值.例如对于标识(001101010),所划分的 3 个二进制位组为(001,101,010),对应的象限序列为(021).表 2 中 shl 含义与定义 7 中 shl 相同,用来判断位置标识和 K 象限序列匹配长度.

L_s 值可以通过位置标识计算得出,方法如下:首先将节点位置标识按 3 个二进制位为一组进行划分,计算分成的组数,在此基础上,如果位置标识的末位为 1,说明该节点为 CSP,层次加 1.如果末尾为 0,则节点为 BSP,层次不再加 1.以图 1 中 B_{12}(001001)和 B_1(001000)为例,由于 B_{12} 位置标识划分为 2 组,并且末位为 1,表明 B_{12} 位于第 3 层,而 B_1 位置标识划分为 2 组,并且末位为 0,所以 B_1 位于第 2 层.

算法 1. Procedure Routing(S,K).

```
1. l=shl(PI_s,K);   //计算象限序列匹配长度
2. if (l==0){       //值为0,表明不在同一顶层象限
3.   S=D_s^k; l=shl(PI_s,K);}
                    //象限转移,定位到位置标识与
                    //K 在同一顶层象限的节点
4. if (S∈CSP){      //S 是中心超级节点 CSP
5.   S=FWD(S,L_s-l-1);}
     //从 CSP 向上传递 L_s-l-1 层,定位到 S 位置标
     //识与 K 象限序列最长交集对应的 CSP
6. else {           //S 是端超级节点 BSP
7.    S=FWD(S,L_s-l);} //从 BSP 传递时多一跳
8. while (l<G_K){
   //从该 CSP 逐层向下定位,也可以从 BSP 向下传
     递,原理类似,这里不再赘述.
9.   if ((S_1=GN(S,G_K^{l+1}|1))!=null){
     //先查找 S 的下层 CSP,|表示位或
10.      S=S_1; l++;}
           //取 K 的下一组 3 个二进制位进行比较
11.  else if ((S_2=GN(S,G_K^{l+1}&0))!=null){
     //查找 S 的同层 BSP,& 表示位与
12.     Target=S_2; Break;}
         //目标节点为同层 BSP
13.  else {//下层 CSP 和同层 BSP 都不存在
14.     Target=S; Break;} //目标节点为 S 自身
15. }
```

定理 2. 在完全 Quad 系统中,从任意一个超级节点 $\overline{S}$ 出发,按照资源 ID 标识(设为 K)定位到目标节点的最大路径长度 $RL_{\max}$ 为 $L_s-2l+ML+1$,平均路径长度 RL_{avg} 为 $\frac{3ML}{2}-\frac{97}{60}$,其中 L_s 为超级节点 S 层次,l 为 $shl(PI_s,K)$,表示 S 位置标识与 K 象限序列匹配长度,ML 为式(2)值 $\log_4\left(\frac{3N}{5}+1\right)$.

证明. 在 Quad 中,按照 Routing 算法,从任一超级节点出发到达目标节点最多经历 3 个阶段:

(1) 象限转移阶段. Routing 算法步 2～3，当 $\overline{S}$ 位置标识与 K 不在同一顶层象限内时，$\overline{S}$ 从象限路由表中查找位置标识与 K 位于同一顶层象限且象限序列匹配最长的超级节点 S，并将请求传递给 S，路径长度 H 为 1 跳(H 是指在象限转移时路由中的跳数，1 跳表示路由过程中从一个节点到下一个节点的一次转移)；如果在同一顶层象限内，S 与 $\overline{S}$ 相同，不需进行转移. 显然象限转移概率为 3/4，象限转移阶段平均路径长度为 3/4.

(2) 上升阶段. Routing 算法步 4～7，在同一顶层象限内从 S 向上定位到位置标识与 K 象限序列最长交集所对应的 CSP. 上升阶段最大长度为 L_s-l.

节点 S 可能是发起查询的超级节点 $\overline{S}$，也可能是 $\overline{S}$ 通过象限路由表查找到的新节点，L_s 在各层的概率 p 是相等的，概率为 $\frac{1}{ML}$，L_s 的平均值 AL_s 为

$$AL_s=\sum_{i=1}^{ML}p_iL_i=\sum_{i=1}^{ML}L_i/ML=\frac{ML\times(ML+1)}{2\times ML}=\frac{(ML+1)}{2} \tag{3}$$

在完全 Quad 拓扑的同一顶层象限内，S 与目标节点的位置标识象限序列在第 1 层匹配的概率为 1，在第 2 层匹配的概率为 1/4，因为第 2 层包含 4 个子象限，在第 3 层匹配的概率为 1/16，依次类推，l 的平均值 Al 表示为 $\sum_{i=1}^{AL_s}\frac{1}{4^{i-1}}$. 在两个超级节点中，目标节点通常在最大层，因此只需比较到 L_s 即可，其平均值 AL_s 为 $\frac{(ML+1)}{2}$，Al 值为

$$Al=1+\frac{1}{4}+\cdots+\frac{1}{4^{AL_s-1}}=\left(1-\frac{1}{4^{AL_s}}\right)\Big/\left(1-\frac{1}{4}\right)\approx\frac{4}{3} \tag{4}$$

式(4)中，当 $AL_s\geq 2$ 时，$1/4^{AL_s}$ 较小；当 $AL_s=1$ 时，$ML=1$，整个系统只有一层，属于特例，最多只需 1 跳就可以直接定位到目标节点.

在上升阶段，从 CSP 向上传递时比 BSP 要少一跳. 由于在完全 Quad 系统的一个象限中同层 CSP 和 BSP 比例为 1∶4，节点是 CSP 的概率为 1/5，因而上升阶段平均路径长度为 $AL_s-Al-1/5$.

(3) 下降阶段. Routing 算法步 8～15，从象限序列最长交集对应的 CSP 定位到目标节点，因为存储资源索引的超级节点层次不超过分层象限空间最大层 ML，所以下降阶段最大路径长度为 $ML-l$.

最大路径长度 $RL_{\max}$ 为

$$RL_{\max}=H+(L_s-l)+(ML-l)=L_s-2l+ML+1 \tag{5}$$

平均路径长度 RL_{avg} 为

$$RL_{\text{avg}}=3/4+(AL_s-Al-1/5)+(ML-Al)=\frac{3}{4}+\frac{(ML+1)}{2}-\frac{4}{3}-\frac{1}{5}+ML-\frac{4}{3}=\frac{3ML}{2}-\frac{97}{60} \tag{6}$$

证毕.

5　自组织、负载均衡与容错

P2P 网络具有很高的动态性，大量节点可能随时加入和退出系统，因而 P2P 网络拓扑需要具有良好的自组织性来处理节点加入和离开，并通过调整、分裂以及合并机制来保持超级节点间的负载均衡，同时拓扑还需要具有容错能力.

5.1　节点加入

新叶节点 F 加入 P2P 网络，需要知道网络中已经存在的任意一个超级节点 S，当 F 与 S 成功建立连接之后，F 将共享资源索引传递给超级节点 S，S 将其加进本地非结构化索引库，并通过结构化路由发布到相关超级节点. 如果 S 接受 F 加入后负载超出其能力限制，就需要进行拓扑调整和分裂.

5.1.1　加入调整过程

超级节点在其能力范围内接受叶节点加入，设超级节点 S_i 的容量为 C_i，当前负载为 D_i，则 S_i 负载率为 $\frac{D_i}{C_i}$，设 α_u 为超级节点负载上限率，即超级节点负载与容量之比的上限要求，超级节点负载率大于该值后，必须将一部分负载转移给其它超级节点，β_u 为调整上限率，当超级节点的负载与容量之比小于该值时才可以接受其它超级节点转移负载. β_u 保证了接受转移的超级节点拥有一定的空闲容量，防止超级节点接受其它超级节点的叶节点后负载率接近 α_u. 进行调整时，负载率大于 α_u 的超级节点 S_i 首先检查同层超级节点，按下列情况进行不同的处理，为清晰起见，我们给出图 3 做辅助说明.

(1) S_i 不存在同层超级节点，则 S_i 检查是否存在上层超级节点 S_p 满足负载 $D_p<\beta_uC_p$，如果有，则 S_i 向 S_p 转移 $\left\lfloor\frac{D_iC_p-D_pC_i}{C_i+C_p}\right\rfloor$ 个节点，使两个超级节点尽可能保持相同的负载率，$\lfloor\ \rfloor$ 表示向下取整. 如果 S_i 无上层超级节点或上层超级节点负载 $D_p\geq\beta_uC_p$，则对 S_i 进行分裂. 例如图 3 中第一顶层象限内超级

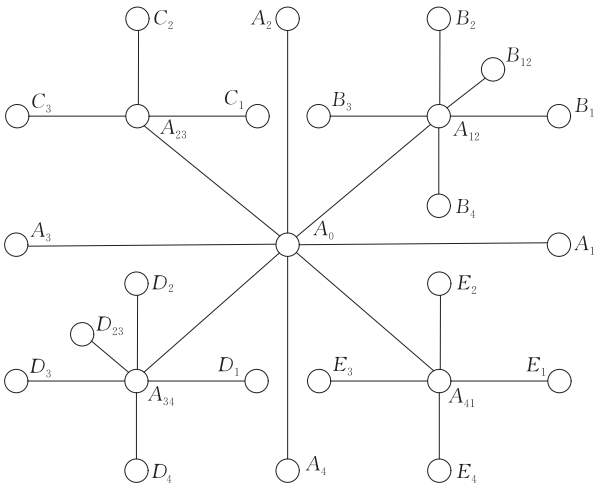

图 3 调整过程

节点 B_{12} 过载时，先检查是否存在同层邻居超级节点，当没有同层邻居节点时，按照 Quad 逐层构造规则，也不会有下层节点，因而 B_{12} 只有检查上层超级节点 A_{12} 和 B_1，为了降低负载转移对拓扑产生的影响，B_{12} 只检查自己的直接上层超级节点，不再向上递归传递. 如果 A_{12} 和 B_1 中负载最轻的节点负载率小于 β_u，则 B_{12} 向其转移负载，如果不满足转移条件，则对 B_{12} 进行分裂，产生新的超级节点.

(2) S_i 存在同层超级节点，并且其中负载最小的 S_j 满足负载 $D_j<\beta_u C_j$，则 S_i 向 S_j 转移 $\left\lfloor\frac{D_iC_j-D_jC_i}{C_i+C_j}\right\rfloor$ 个节点. 如图 3 中第二顶层象限内的超级节点 C_1 过载时，检查同层超级节点邻居 A_{23}、C_2 和 C_3，如果负载最轻的节点负载率小于 β_u，则 C_1 向其转移负载.

(3) S_i 存在同层超级节点，并且其中负载最小的节点 S_j 负载 $D_j\geq\beta_u C_j$，不能进行转移，则检查上层邻居，如果其负载率低于调整上限率 β_u，则向其转移相应的负载，如果上层邻居不满足转移条件，S_i 检查下层邻居，按照下列情况进行相应处理：

① S_i 无下层超级节点，则对 S_i 进行分裂. 例如图 3 第二顶层象限内的超级节点 C_2 过载时，其同层邻居 A_{23}、C_1 和 C_3 和上层邻居 A_2 负载率均不小于 β_u，不满足调整条件并且由于 C_2 无下层超级节点，所以对其进行分裂.

② S_i 存在下层超级节点，并且其中负载最小的超级节点 S_k 满足 $D_k<\beta_u C_k$，则 S_i 向 S_k 转移 $\left\lfloor\frac{D_iC_k-D_kC_i}{C_i+C_k}\right\rfloor$ 个节点. 如第三顶层象限内的 D_2 过载，其同层邻居 A_{34}、D_1、D_3 和 D_4 和上层邻居 A_4 负载率不小于 β_u，不满足调整条件，但下层邻居 D_{23} 负载率小于 β_u，则 D_2 向 D_{23} 转移负载.

③ S_i 存在下层超级节点，并且其中负载最小的超级节点 S_k 负载 $D_k\geq\beta_u C_k$，则 S_i 向 S_k 发送调整消息，S_k 递归执行步(3)，如果 S_k 仍然不能完成调整，则需要继续向下层传递，直到调整结束. 如图 3 中第四顶层象限内的 A_4 过载，同层 A_0、A_1、A_2、A_3 负载率均不小于 β_u，则 A_4 检查下层超级节点 A_{41}、E_1、E_2、E_3、E_4，如果其中负载率最轻的节点比如 E_2 负载率不小于 β_u，则 A_4 将转移请求发送给 E_2、E_2 执行步(3)，递归进行调整.

5.1.2 分裂过程

在节点不断加入的过程中，Quad 拓扑会自适应的进行调整，新的超级节点会随着分裂不断增加. 图 4 显示了二层 Quad 拓扑的动态分裂过程，为清晰起见，(c)、(d)、(e) 中心超级节点的叶节点被省略.

在分裂过程中，系统中的第 1 个节点自动成为超级节点，该超级节点在其能力范围内接受叶节点加入，如图 4(a) 所示，当叶节点数超出其能力限制并且无法通过调整转移给其它超级节点时，超级节点向第一象限的 0 方向分裂出一个新超级节点，如图 4(b) 所示，在随后的分裂过程中，超级节点在其它未分配的 (2，4，6) 3 个方向继续分裂新超级节点，如图 4(c). 当一层的方向分配完后，超级节点向下层方向分裂，如图 4(d). 由于超级节点在分裂之前，会先在同层超级节点进行调整，所以只有在同层象限都满的情况下才向下层分裂，不会出现 4 个同层象限邻居还有位置空间的情况向下层分裂. 另外，通过向上层超级节点调整负载，一个象限的负载会间接转移到上层其它象限. 二层 Quad 拓扑最终分裂后的情况如图 4(e) 所示.

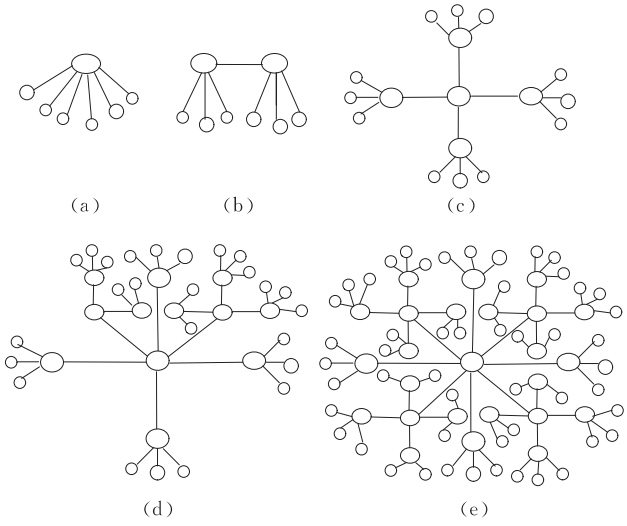

图 4 动态分裂过程

设需要分裂的超级节点为 S_i，具体分裂步骤如下：

(1) 确定分裂方向. 超级节点分裂方向遵循层次优先、象限顺序优先原则. 节点首先判断邻居路由表中同层(0,2,4,6) 4 个方向中是否有未分配的索引，如果有，则选择标号最低的方向. 如果同层 4 个方向已满，则从下层超级节点确定方向，其方向确定规则是：如果 S_i 是 CSP，则分别按 4 个象限子中心超级节点($CCSP$)索引顺序中未分配的方向确定分裂方向，对应邻居路由表索引(1,3,5,7)，如果 S_i 是 BSP，则以 S_i 对应的下层 $CCSP$ 以及 $CCSP$ 的同层(0,2,4,6)方向为序确定分裂方向，分别对应 S_i 的邻居路由表索引(8,1,3,5,7).

(2) 选择新超级节点 S_j. S_j 从 S_i 的候选超级节点产生，候选超级节点是叶节点中最优秀的节点. 新超级节点的位置标识由距离其最近的 CSP 位置标识加分裂方向构成.

(3) 叶节点转移. S_i 转移 $\left\lfloor \frac{D_iC_j}{C_i+C_j} \right\rfloor$ 个叶节点给新超级节点 S_j.

(4) 路由表和索引处理. 按照分裂方向，S_i 和 S_j 分别添加邻居路由表索引. S_j 象限路由表先从 S_i 复制，S_j 再通过这些节点获得层次更为接近的超级节点. 在索引处理上，S_i 把与 S_j 象限序列相匹配的结构化倒排索引表表项转移给 S_j. 转移时遵循同层 CSP 或 BSP 转移给同层 BSP，上层 BSP 转移给下层 CSP 原则. 非结构化资源索引不进行转移，由相关超级节点从叶节点中重新创建.

5.2　节点退出

节点可能因为多种原因离开 P2P 网络，比如节点选择主动离开或因为网络故障被动离开. 主动离开情况下，节点可以通知其它节点完成相应拓扑调整，被动离开时，其邻居节点通过定期心跳函数才可以检测到，处理成本增加并具有滞后性，但两者处理方式比较类似，为简化起见，本文对这两种情况不做进一步区分.

节点分为超级节点和叶节点，其退出对拓扑有不同影响. 超级节点退出会对路由产生影响，因而需要修复拓扑，方法是由候选超级节点替代退出的超级节点. 候选超级节点位置标识设为退出超级节点的位置标识，并与其邻居路由表和象限路由表中的超级节点建立连接，然后候选超级节点接受退出超级节点的叶节点作为子节点并从中选择新候选超级节点.

叶节点退出时，超级节点需要将其从本地非结构化索引中删除，叶节点的结构化索引不作处理，因为超级节点会定期对倒排索引表中的叶节点发 PING 消息，如果没有收到 PONG 消息，会将未响应节点的索引项从倒排索引表中删除. 设 α_b 为负载下限率，即当超级节点负载与容量之比低于该比率时，超级节点必须从其它超级节点移进一部分负载；β_b 为调整下限率，即超级节点的负载与容量之比不低于该值时才可以向其它超级节点转移负载. 当叶节点退出使父超级节点负载率小于负载下限率 α_b 时，触发退出调整和合并行为.

5.2.1　退出调整过程

退出调整与加入调整过程类似，负载低于 α_b 的超级节点 S_i 检查同层超级节点，按下列情况进行不同的处理：

(1) S_i 不存在同层超级节点，则检查 S_i 是否存在上层超级节点 S_p 满足负载 $D_p \geq \beta_b C_p$，如果存在，则 S_p 向 S_i 转移 $\left\lfloor \frac{D_pC_i - D_iC_p}{C_i+C_p} \right\rfloor$ 个节点，如果 S_i 无上层超级节点或上层超级节点负载 $D_p < \beta_b C_p$，则由 S_i 发起合并过程.

(2) S_i 存在同层超级节点，并且其中负载最大的 S_j 满足负载 $D_j \geq \beta_b C_j$，则 S_j 向 S_i 转移 $\left\lfloor \frac{D_jC_i - D_iC_j}{C_i+C_j} \right\rfloor$ 个节点.

(3) S_i 存在同层超级节点，并且其中负载最大的超级节点 S_j 负载 $D_j < \beta_b C_j$，不满足转移条件，则继续检查上层邻居，如果其中负载最大的上层超级节点负载率不低于调整下限率 β_b，则向 S_i 转移相应的负载，如果上层邻居不满足转移条件，则 S_i 需要进一步检查下层超级节点，按照下列情况进行相应的处理.

① S_i 无下层超级节点，则对 S_i 进行合并操作.

② S_i 存在下层超级节点，并且其中负载最大的 S_j 满足 $D_j \geq \beta_b C_j$，则 S_j 向 S_i 转移 $\left\lfloor \frac{D_jC_i - D_iC_j}{C_i+C_j} \right\rfloor$ 个叶节点.

③ S_i 存在下层超级节点，并且其中负载最大的超级节点 S_k 负载 $D_k < \beta_b C_k$，则 S_i 向 S_k 发出调整消息，S_k 递归执行步(3)，如果 S_k 不能在本层完成调整，需继续向下层传递，直到调整结束.

5.2.2　超级节点合并

当超级节点自身的负载无法通过调整满足最低限制要求时，需要进行合并. 超级节点合并过程与分

裂过程大体相反. 在合并方向上,遵循同层 *BSP* 向同层 *CSP* 合并,下层 *CSP* 向上层 *BSP* 合并规则. 超级节点需要调整路由表中与被合并超级节点对应的表项. 在叶节点转移上,被合并超级节点和其包含的叶节点一起转移给发起合并的超级节点. 在索引处理上,被合并节点的倒排索引表转移给合并方向的超级节点. 非结构化索引部分由合并后的超级节点从叶节点中重新创建.

5.3 拓扑容错机制

P2P 网络工作在一个具有高度动态性和成员不可靠的环境下,因而 P2P 系统必须具备有效的容错机制来保证超级节点失效后,网络仍然能够正常运行. Quad 具有周期性检测和冗余路径两种容错方法,候选超级节点会定期检测父超级节点是否处于正常状态,如果其失效,候选超级节点会很快接替失效的超级节点来修复拓扑. 在修复期间,Quad 通过冗余路径保证其它超级节点之间的正常路由. 从同层超级节点来看,每个超级节点都包含同层其它邻居超级节点信息,*CSP* 或 *BSP* 失效不影响其它超级节点之间的互连. 从上下层超级节点来看,邻居超级节点间的路径具有冗余设计,每一对上下层超级节点间都有两条以上路径,如图 5 所示.

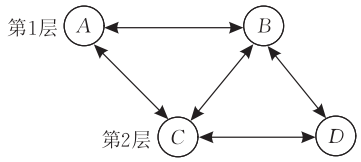

图 5 邻居路由表冗余设计

图 5 显示了一个象限的两层超级节点之间关系图,其中超级节点 *A* 和 *B* 是第 1 层 *CSP* 和 *BSP*,*C* 是 *A* 和 *B* 的下层中心超级节点(*CCSP*),*D* 是 *B* 的下层端超级节点(*CBSP*),可以看出,图中任意两个节点间都有两条以上路径,如 *A* 可以通过 *B* 到达 *D*,也可以通过 *C* 到达 *D*,图中任何一个超级节点或一条路径失效都不影响其它超级节点之间的连通性. 当*B*和*C*同时失效时,只要有一个候选超级节点修复完成,*A* 和 *D* 之间就可以进行路由. 如果一个象限的所有超级节点如 *C* 和 *D* 都失效,失效节点的候选超级节点可以通过预先备份的邻居路由表找到上下层超级节点并修复拓扑. 如果同一象限相邻两层所有超级节点同时失效,失效超级节点的候选超级节点可以通过象限路由表向其它象限的超级节点进行查询,获得修复拓扑所需的本象限其它各层超级节点信息,避免网络被分割.

6 分 析

6.1 与结构化超级节点的拓扑比较

由于 IS-P2P 网络模型[13]在超级节点层采用了典型的结构化系统,我们将 Quad 与 IS-P2P 模型中提出的 4 个系统进行比较,比较包括 3 个方面:(1)查找路径长度. 定义了从任意一个超级节点出发,到达目标节点的跳数;(2)路由状态数. 路由状态数反映了超级节点路由表以及相关集合中的节点数量,路由状态数越大,拓扑维护费用就越高. 一般来说,结构化系统的查找路径长度和路由状态数之间存在一个渐近曲线关系,路由状态数越大,查找路径越短,反之亦然;(3)节点加入或离开产生的最大拓扑修复成本. 我们取两者中的最大值作为比较参数.

Quad 和 IS-P2P 模型中 4 个结构化系统比较结果如表 3 所示. 在 5 种结构的超级节点层,IS-Chord 采用了带弦环结构,IS-CAN 为多维空间结构,IS-Pastry、IS-Tapestry 是超立方体结构,Quad 采用分层象限空间结构. 从查找路径长度来看,5 个系统的路由时间复杂度均为 $O(\log N)$. 表 4 显示了 5 个系统的查找路径长度和路由状态数在不同超级节点规模下的计算结果. 由于 Quad 采用了八进制,为了保证维数或进制的一致,我们除 IS-Chord 仍采用经典的二阶外,IS-CAN 采用三维,IS-Pastry 和 IS-Tapestry 取八进制作为参数,另外,IS-Pastry 的 $|L|$ 和 $|M|$ 值取 B 作为参数.

表 3 Quad 和结构化超级节点系统比较

系统	参数	查找路径长度	路由状态数	加入或离开最大拓扑修复数
IS-Chord	N:超级节点数,下同	$\log_2 N$	$\log_2 N$	$(\log_2 N)^2$
IS-CAN	d:维数	$\frac{d}{4}\cdot N^{1/d}$	$2\cdot d$	$2\cdot d$
IS-Pastry	B:ID 进制,$\|L\|$:叶集大小,$\|M\|$ 邻居集大小	$\log_B N$	$(B-1)\cdot\log_B N+\|L\|+\|M\|$	$B\cdot\log_B N$
IS-Tapestry	B:ID 进制	$\log_B N$	$B\cdot\log_B N$	$B\cdot\log_B N$
Quad	ML:$\log_4\left(\frac{3N}{5}+1\right)$	$\frac{3ML}{2}-\frac{97}{60}$	16	16

表 4　Quad 和结构化超级节点系统在不同超级节点规模下数据值比较

系统	超级节点为 1000 个时的数值		超级节点为 10000 个时的数值		超级节点为 100000 个时的数值		超级节点为 1000000 个时的数值	
	路径长度	状态数	路径长度	状态数	路径长度	状态数	路径长度	状态数
IS-Chord	10	10	13.3	13.3	16.6	16.6	19.9	19.9
IS-CAN	7.5	6	16.2	6	34.8	6	75	6
IS-Pastry	3.3	39.3	4.4	47	5.5	54.8	6.6	62.5
IS-Tapestry	3.3	26.6	4.4	35.4	5.5	44.3	6.6	53.2
Quad	5.3	16	7.8	16	10.3	16	12.8	16

从查找路径长度来看，IS-Pastry 和 IS-Tapestry 最短，Quad 次之，然后是 IS-Chord 和 IS-CAN. 从查找路径长度增长趋势来看，随着超级节点规模增大，IS-Pastry 和 IS-Tapestry 的查找路径长度增幅仍然最小，Quad 和 IS-Chord 均保持平缓的增长，IS-CAN 在这方面最差，因为当超级节点规模增加时，IS-CAN 查找路径长度增长幅度远大于其它 4 个系统.

从路由状态数来看，IS-Pastry 和 IS-Tapestry 最大. IS-Chord 在超级节点规模较小时，路由状态数要比 Quad 小一些，但是随着超级节点规模增加，该值会逐渐增加并超过 Quad. IS-CAN 和 Quad 的路由状态数具有 Viceroy、Koorde 和 Cycloid 等常数度数结构化系统的优点，在不同网络规模下都能保持不变，具有较好的可扩展性. 从数值上看，IS-CAN 的路由状态数比 Quad 要小，但代价是随着超级节点规模增大，查找路径长度增长过快.

从超级节点加入或离开产生的最大拓扑修复成本来看，IS-Chord 最高，为路由状态数的平方. 在 IS-Pastry 和 IS-Tapestry 中，因为超级节点加入时要构建路由表并且两者的路由表状态数都较大，所以构造成本要高于 Quad 和 IS-CAN，此外，IS-Chord、IS-Pastry 和 IS-Tapestry 的拓扑修复成本都随超级节点规模增加而增大. 相比之下，Quad 和 IS-CAN 中超级节点加入或离开对拓扑影响小，两者的修复成本与路由状态数相一致，均为常数值，能较好适应拓扑的动态性.

在结构化超级节点拓扑中，超级节点失效不仅会影响路由表，而且会影响其代理的叶节点和存储结构化资源的倒排索引表. 在 Quad 中，由于超级节点在拓扑中的位置是由其位置标识决定的，独立于 ID 标识，并且位置标识可以赋给从其叶节点中选出的候选超级节点，所以候选超级节点可以自然替代失效的超级节点，其它叶节点只需将候选超级节点作为父超级节点即可，不需要寻找新超级节点加入. 此外，通过在候选超级节点预先备份超级节点的倒排索引表，超级节点失效后原有的结构化资源索引不受影响. 而在 IS-Chord、IS-CAN、IS-Pastry 和 IS-Tapestry 4 个结构化超级节点系统中，超级节点只有 ID 标识，没有位置标识，因为不同 ID 标识的超级节点在拓扑中的位置不一样，因而即使采用候选超级节点替代机制解决叶节点重新连接问题，失效超级节点的结构化倒排索引表也不能直接转移给候选超级节点，该表要么通过候选超级节点转移到与失效超级节点标识相近的超级节点，要么事先将倒排索引表备份到标识相近的超级节点，或者系统采用多个 Hash 函数进行映射以实现冗余备份，但无论上述哪种方法，都会带来额外的路由和存储成本，相比之下，Quad 中超级节点失效产生的资源索引修复成本是最小的.

通过上述分析，Quad 与结构化超级节点拓扑相比具有以下特征：

(1) Quad 在查找路径长度和路由状态数上保持了更好的均衡性.

(2) Quad 具有常数度数结构化系统的特征，在每个超级节点维护固定路由表项数的同时，仍然能达到 $O(\log N)$ 跳的定位效率.

(3) Quad 在超级节点失效修复成本方面具有优势，只需较低的修复成本就能快速恢复拓扑，并且不影响倒排索引表，能更好地适应动态环境.

6.2　Quad 拓扑结构的负载均衡性

结构化超级节点拓扑的负载均衡机制与单层的结构化拓扑具有很大的不同. 在单层结构化拓扑中，一致性 Hash 函数保证了映射后的 ID 标识具有良好的均衡性，在此基础上构建的拓扑也是均衡的，但是在结构化超级节点拓扑中，每个超级节点都要代理一定数量的叶节点，由于叶节点加入和退出具有随机性，因此即使 ID 标识是均衡分布的，各个超级节点的负载也不能自动实现均衡，而不平衡的拓扑会带来路由效率低下，系统容易形成热点等诸多不足.

Quad 在负载均衡方面具有自适应的调整、分裂与合并机制，与 IS-Chord、IS-CAN、IS-Pastry 和 IS-Tapestry 相比，Quad 除了可以向同层邻居超级节点(类似于 IS-CAN 相邻区域的超级节点)调整负载，还可以向上下层超级节点(类似于 IS-Chord、

IS-Pastry 和 IS-Tapestry 的前驱与后继超级节点)转移叶节点.在调整顺序上,先是在同层超级节点之间进行调整,保证了同一层次的各个象限保持均衡.而上下层超级节点间的负载调整能使下层区域的负载逐步转移到上层其它区域.另外,在向上层超级节点调整的时候,只考虑直接上层超级节点,而不是递归向上,使一次调整不会对象限其它区域产生大的影响.

在分裂与合并方面,因为 Quad 中新超级节点在拓扑中的位置标识(PI)与其分裂或合并方向相对应,分裂与合并只在本区域进行,不影响其它区域,Quad 这种特性也降低了其物理网络拓扑和覆盖网(Overlay)不匹配问题,因为节点总是就近进行处理.而在 IS-Chord、IS-CAN、IS-Pastry 和 IS-Tapestry 中,ID 标识决定了超级节点在拓扑结构中的位置,如果新产生的超级节点 ID 标识对应于其它拓扑位置,或者超级节点不能和相邻区域的超级节点合并时,分裂与合并就要跨区域进行,对拓扑影响较大.通过以上分析,Quad 在负载均衡方面具有以下特点:

(1)超级节点数量能够随系统中节点数量的增加或减少作自适应的增减.

(2)即使大量节点向少数超级节点请求加入,Quad 也能够将这些节点均衡的调整到不同层次的不同象限区域.

(3)调整、分裂与合并机制不会引起拓扑"雪崩效应",即少量拓扑调整不会引起很大的拓扑变化.

6.3 Quad 结构化路由性能的改进

Quad 结构化路由分为象限转移、上升和下降 3 个阶段,通过对路由表进行调整,并利用缓存就可以有效改进其路由性能,以下是两种改进 Quad 路由性能的方法:

(1)利用顶层象限内同层超级节点"短链",减小上升阶段的长度.在 Quad 中,象限路由表只能解决不同顶层象限之间的超级节点定位,不能实现同一顶层象限内的不同子象限同层超级节点定位,我们称前者为"长链",后者为"短链".在式(4)中,我们证明了"长链"可以使象限序列匹配平均值达到 4/3,因而很多查询需要上升到接近第 2 层才能向下定位,不仅增加了路由长度,也会增加这些节点的压力.通过在路由表中增加"短链"可以有效解决这个问题.在 Quad 一个顶层象限内,第 2 层包括最多 5 个超级节点,这些节点利用邻居路由表可以完成互相定位,其作用相当于"短链".在一个顶层象限内的第 3 层最多包括 20 个超级节点,分属于 4 个不同的第 2 层子象限,所以第 3 层一个象限内的超级节点只需要 3 个"短链"节点就可以定位到其它 3 个属于不同第 2 层子象限的第 3 层象限,在这些第 3 层象限中,"短链"节点通过邻居路由表可以找到和资源 ID 象限序列匹配长度为 3 的超级节点.同理,在一个顶层象限内的第 4 层最多包含 80 个超级节点,属于 16 个第 3 层子象限,第 4 层的超级节点只需要 12 个"短链"节点就可以定位到属于其它 12 个第 3 个象限的第 4 层超级节点(其它 3 个第 3 层象限的第 4 层子象限可直接利用邻居路由表定位),这些"短链"节点通过邻居路由表可以找到和资源 ID 象限序列匹配长度为 4 的超级节点.因此我们只需在第 3 层超级节点设置 3 个"短链"节点,在第 4 层超级节点设置 12 个"短链"节点(其它层不需设"短链")就可以显著提高路由效率,并减小上层节点压力,因为象限序列匹配值 l 会对路由长度产生 $2l$ 影响,并且 Quad 拓扑第 4 层最多可包含 320 个超级节点,平均路由压力不大.如果只设置第 3 层超级节点的 3 个"短链",则不需增加 Quad 路由表数量,只需将象限路由表数组项中的一个节点变成"短链"(需要增加"短链"节点象限标识).此时,上升阶段最多只需要上升到第 3 层即可,不仅缩短了上升阶段长度,而且第 2 层只需承担很小的路由任务.而在 Quad 拓扑的第 3 层,超级节点数量最多可达 80 个,也可以有效缓解超级节点路由压力.

(2)利用结构化索引缓存,减小下降阶段的长度.在 Quad 中,随着各层超级节点的不断分裂,结构化索引不断被转移到更低层的超级节点.所以在路由下降阶段总是要到达最低层才可以定位到存放结构化索引的目标节点.如果在结构化索引向下转移过程中,上层超级节点缓存一定数量的结构化索引副本,则定位过程不一定要下降到最低层超级节点,在下降阶段的过程中就有可能通过倒排索引表副本查找到资源信息.

7 模拟验证

因为非结构化超级节点拓扑在拓扑构造和失效修复等方面比结构化超级节点拓扑更具成本优势,所以本文将 Quad 和其中较优的非结构化超级节点拓扑进行比较.我们在 PeerSim① 基础上用 Java 分别实现了 Quad 和类似于 Gnutella 0.6 版的两层非结构化超级节点拓扑(简称为 Gnu)模拟程序,比较

① PeerSim Simulator. http://peersim.sourceforge.net/

内容包括拓扑构造成本、调整和分裂数、失效修复成本 3 个方面，另外我们也模拟验证了 Quad 负载均衡特征. 模拟中两种协议的节点构造方式、随机函数都保持一致，节点容量服从 Power-Law 规则，分布指数为 2.2，具体参数如表 5 所示.

表 5　模拟参数

参数	含义	值
N	网络规模	5000～40000
F	失效比率	0.3～0.8
C	节点容量	20～80
D	节点负载	20～80
α_u	负载上限率	0.9
β_u	调整上限率	0.8
α_b	负载下限率	0.2
β_b	调整下限率	0.3

7.1　拓扑构造成本

在 Quad 拓扑构造中，节点之间的交互通过消息传递来实现，我们用传递的消息数来表示拓扑构造成本，具体包含以下 3 类：

(1) 请求成本，节点向其它节点请求加入产生的消息数.

(2) 接受成本，节点响应其它节点的加入请求产生的消息数.

(3) 移动成本，叶节点在超级节点之间移动产生的消息数.

由于请求成本和接受成本是相对应的，因此我们在模拟中只考虑接受成本和移动成本. Quad 和 Gnu 拓扑构造产生的消息成本如图 6 和图 7 所示. 图 6 比较了不同规模新节点加入网络时产生的总接受成本和总移动成本，可以看出 Quad 在这两个方面均低于 Gnu. 例如当 20000 个新节点加入时，Quad 总接受成本是 45145，总移动成本是 25146，而 Gnu 的总接受和总移动成本分别为 49742 和 29743；当 40000 个新节点加入时，Quad 总接受成本是 80734 而 Gnu 是 94243，总移动成本方面 Quad 是 45735，小于 Gnu 的成本 54244.

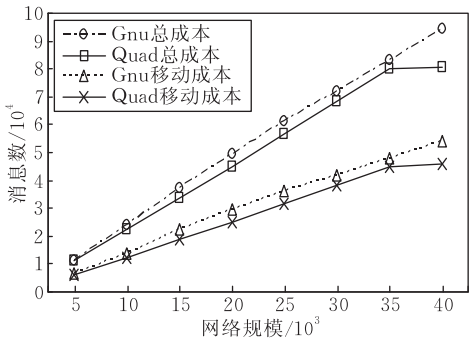

图 6　消息总成本比较

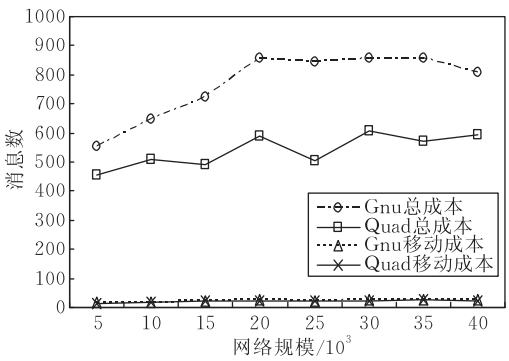

图 7　单个节点最大消息成本比较

图 7 比较了单个节点产生的最大移动成本和接受成本. Quad 中单个节点最大移动成本在不同网络规模下的平均值要比 Gnu 低 1.5 个消息数. 在单个节点最大接受成本方面，Quad 要明显优于 Gnu，如 40000 新节点加入时，前者成本值为 592，后者为 807，这表明和 Gnu 相比，Quad 中节点成为热点的可能性要小一些.

7.2　调整与分裂数

为了保证超级节点之间的负载均衡，超级节点会通过调整和分裂转移自身负载，调整是在超级节点间转移叶节点负载，会增加超级节点的消息处理成本并影响其本地非结构化资源索引，而分裂会产生新的超级节点，影响超级节点路由表和资源索引，因而在保证拓扑均衡基础上减少调整和分裂数有利于降低系统负载. 图 8 对 Quad 和 Gnu 拓扑调整和分裂情况进行了比较，从图中可以看出，Quad 和 Gnu 在分裂次数上非常接近，超级节点数占总节点数比例均保持在 1.6%～1.8%之间. 在调整次数上，Quad 比 Gnu 要低很多，例如当加入节点数为 20000 时，Quad 和 Gnu 的调整数分别为 2502 和 7333，当加入节点数为 40000 时，Quad 和 Gnu 的调整数为 5309 和 17223，Quad 调整次数只占 Gnu 的三分之一左右. 这表明，在保持拓扑负载均衡上，Quad 不需要进行频繁的叶节点转移.

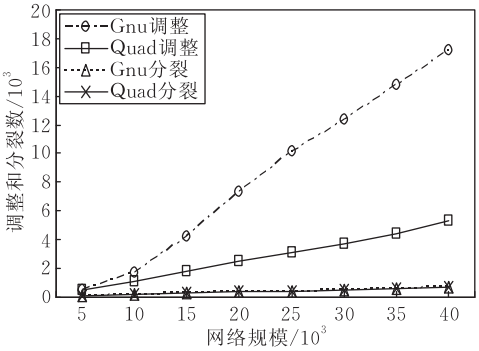

图 8　调整和分裂数

7.3 失效修复成本

由于 P2P 网络具有高度动态性,能否有效处理节点失效对系统性能有重要影响.图 9 和图 10 显示了网络规模为 40000 时,超级节点和叶节点在不同失效比例下的修复成本.图 9 显示了超级节点失效情况,可以看出,Quad 在不同失效比例下的修复成本要远小于 Gnu,比如当超级节点失效比率为 30% 时,Quad 和 Gnu 修复成本分别为 13119 和 40209,当失效比率为 80%,Quad 修复成本为 35077,而 Gnu 修复成本多达 108999,Quad 的修复成本只占 Gnu 的 32%左右,表明 Quad 中超级节点失效对拓扑影响较小.

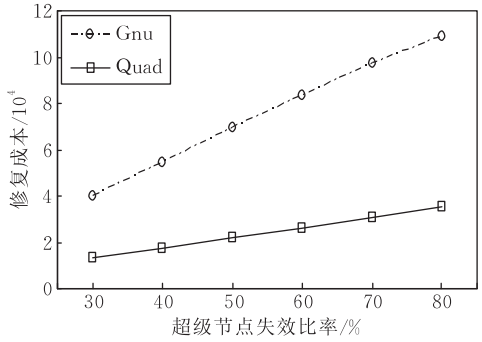

图 9 超级节点失效修复成本

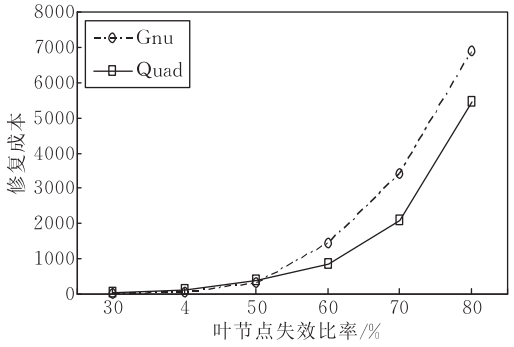

图 10 叶节点失效修复成本

当叶节点失效或退出使超级节点负载小于负载下限率 α_b 时,会引起超级节点之间的调整和合并.图 10 显示不同叶节点失效比例下的调整和合并成本.可以看出,当失效比例较低时,Quad 和 Gnu 拓扑修复成本大体接近,但当失效比例持续增加时,Gnu 失效修复成本要明显超过 Quad,例如当叶节点失效比例为 60%时,Quad 和 Gnu 的失效修复成本分别为 834 和 1442,当叶节点失效比例为 80% 时,两者的修复成本分别为 5434 和 6890.这表明 Quad 虽然具有结构化特征,但在处理节点失效方面仍要优于 Gnu.

7.4 负载均衡

在 Quad 拓扑中,当超级节点过载时,能将叶节点转移给负载较轻的超级节点,负载过低时,也能从其它超级节点移进叶节点.通过分裂、调整和合并机制,各个层次和各个象限的超级节点负载能保持相对均衡.

图 11 显示了 Quad 在不同网络规模下各个层次超级节点负载情况.由图可知,各层的超级节点负载率在 0.5 到 0.88 之间,其中 1 层到 3 层的负载比率非常接近,都在 0.8 左右,4 层在 0.75 左右,5 层的超级节点负载率维持在 0.65 左右,6 层由于在最低层,超级节点比其它层多一些空闲空间,超级节点负载率为 0.51.此外,在不同的网络规模下,各层负载比率变化不大,体现了 Quad 拓扑具有良好的负载均衡性.

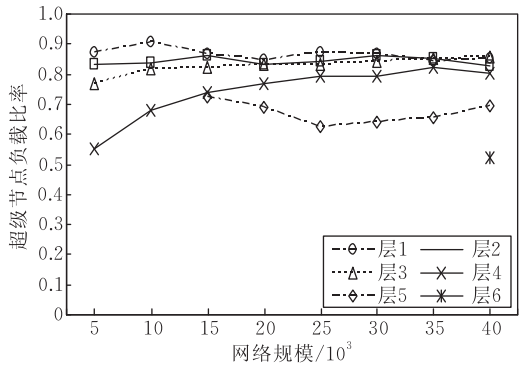

图 11 不同层超级节点负载比率

图 12 比较了 Quad 拓扑在不同象限的超级节点负载情况,数据表明 4 个象限的超级节点负载率差距保持在 10%之内,如网络规模在 20000 时,4 个象限的超级节点负载率差别不到 3.2%,同时各个象限超级节点负载比率在不同网络规模下也保持了较好的稳定性,差异在 5%之内.

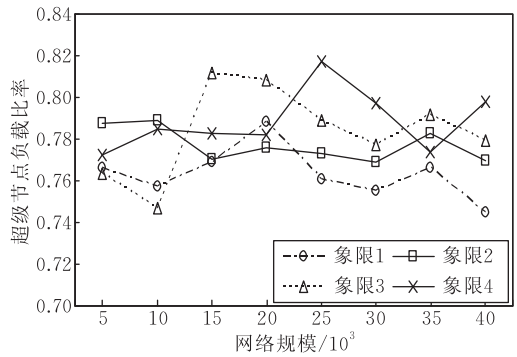

图 12 不同象限超级节点负载比率

8 结束语

本文实现了一种新型超级节点拓扑 Quad,其创新点在于:(1)将分层象限空间应用于 P2P 超级节

点拓扑构造，在保持常数度数下达到 $O(\log N)$ 定位性能，并在定位时间和路由表状态空间方面获得更好的权衡.(2)解决了结构化超级节点拓扑对动态性支持不好的缺点，通过位置标识降低了超级节点失效带来的影响.(3)同时支持数据定位和模糊查询.(4)拓扑具有良好的容错和负载均衡机制，并具有较低的构造和维护成本，提高了应用性能.目前在 P2P 应用方面，结构化系统因为不适应高度动态环境和不支持模糊查询而难以大规模部署于广域网环境.Quad 通过上述改进，能够更好地推动超级节点结构在 P2P 文件共享及流媒体等领域的应用.在下一步工作中，我们将继续改善 Quad 在安全和路由等方面性能，并提高其语义聚类能力，使 Quad 具有更好的实用性.

参　考　文　献

[1] Clarke I, Sandberg O, Wiley B, Hong T W. Freenet: A distributed anonymous information storage and retrieval system//Proceedings of the Workshop on Design Issues in Anonymity and Unobservability. Berkeley, CA, 2000: 46-66

[2] Zhang Yi-Ming, Lu Xi-Cheng, Zheng Qian-Bing, Li Dong-Sheng. An efficient search algorithm for large-scale P2P systems. Journal of Software, 2008, 19(6): 1473-1480(in Chinese)
(张一鸣，卢锡城，郑倩冰，李东升. 一种面向大规模 P2P 系统的快速搜索算法. 软件学报，2008，19(6)：1473-1480)

[3] Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for Internet applications//Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SigComm). San Diego, CA, 2001: 149-160

[4] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network//Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SigComm). San Diego, CA, 2001: 161-172

[5] Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems//Proceedings of the IFIP/ACM International Middleware Conference. Heidelberg, Germany, 2001: 329-350

[6] Zhao B, Kubiatowicz J, Joseph A. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Computer Science Division, University of California, Berkeley: Technical Report UCB/CSD-01-1141, 2001

[7] Maymounkov P, Mazires D. Kademlia: A peer-to-peer information system based on the xor metric//Proceedings of the 1st Internation Workshop on Peer-to-Peer Systems (IPTPS'02). Cambridge, MA, 2002: 53-65

[8] Malkhi D, Naor M, Ratajczak D. Viceroy: A scalable and dynamic emulation of the butterfly//Proceedings of the Principles of Distributed Computing (PODC 2002). Monterey, CA, 2002: 183-192

[9] Kaashoek M F, Karger R. Koorde: A simple degree optimal distributed hash table//Proceedings of the 2nd International Workshop on P2P Systems (IPIPS'03). Berkeley, CA, 2003: 98-107

[10] Chen Gui-Hai, Xu Cheng-Zhong, Shen Hai-Ying, Ye Mao, Liu Zhi-Yu. A new constant-degree P2P overlay network. Chinese Journal of Computers, 2005, 28(7): 1084-1095(in Chinese)
(陈贵海，须成忠，沈海英，叶懋，刘之育. 一种新的常数度数的 P2P 覆盖网络. 计算机学报，2005，28(7)：1084-1095)

[11] Zhang R M, Hu Y C. Assisted Peer-to-Peer Search with Partial Indexing. IEEE Transactions on Parallel and Distributed Systems, 2007, 18(8): 1146-1158

[12] Mizrak A T, Cheng Y C, Kumar V, Savage S. Structured superpeers: Leveraging heterogeneity to provide constant-time lookup//Proceedings of the 3rd IEEE Workshop on Internet Applications. San Jose, California, 2003: 104-111

[13] Xia Qi-Zhi, Xie Gao-Gang, Min Ying-Hua, Li Zhong-Cheng. IS-P2P: Index-Based structured P2P networks. Chinese Journal of Computers, 2006, 29(4): 602-610(in Chinese)
(夏启志，谢高岗，闵应骅，李忠诚. IS-P2P：一种基于索引的结构化 P2P 网络模型. 计算机学报，2006，29(4)：602-610)

[14] Burton H B. Space/Time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970, 13(7): 422-426

FENG Jin-Xiao, born in 1976, Ph. D. candidate. His research interests include peer-to-peer networks and information security.

CHEN Gui-Hai, born in 1963, professor, Ph. D. supervisor. His research interests include parallel computing and wireless networks.

XIE Jun-Yuan, born in 1961, professor, Ph. D. supervisor. His research interests include artificial intelligence and information security.

Background

Topology construction is one of the most essential issues in peer-to-peer (P2P) research field. The earlier P2P topology structures such as central index topology, distributed unstructured P2P topology and distributed structured P2P topology all have some drawbacks. In recent years, researchers proposed the super-peer P2P topology structure which is an enhancement mechanism to the earlier P2P topology and can take advantage of peer capability heterogeneity. The current super-peer topology generation methods consist of unstructured super-peer topology and structured super-peer topology. Although they have improved the routing performance and made topology more stable compared with former distributed unstructured and structured topology structures, they can not solve their root problems. The search results in unstructured super-peer topology are uncertain and structured super-peer topology still does not support proximity search and has high maintenance cost under high dynamic environment. The paper focuses on P2P super-peer topology construction and proposes a novel super-peer topology structure called Quad.

Quad combines the advantages of both unstructured super-peer topology and structured super-peer topology and has the following novel features. First, Quad applies the hierarchical quadrant space technique to P2P super-peer topology construction. Each position in hierarchical quadrant space has unique position identifier and super-peers are organized into these positions. It offers O(logN) routing performance with constant degree and attains a better tradeoff between locating efficiency and routing state size. Meanwhile it avoids the deep impact brought by super-peer failure because the quadrant position identifier of super-peer can be reallocated to the new super-peer in the same position when it fails. Second, hierarchical quadrant space is similar to tree structure and supports DFS and BFS search. Meanwhile hierarchical quadrant space is a digital space, so Quad can support both proximity search and object ID locating. Third, Quad has good load balance and resilience mechanism. The former includes adaptive adjusting, splitting and merging methods while the latter consists of periodical detection and redundancy routing mechanism. Quad also keeps lower topology construction and maintenance cost. Analysis and simulation have shown the good performance of Quad in routing efficiency, topology construction, failure repair and load balance.

This research is partly supported by the National Fundamental Research Program of China (973 program) under grant No. 2006CB303000 and the National Natural Science Foundation of China under grant Nos. 60721002 and 60825205.