用 Small-World 设计无组织 P2P 系统的路由算法*

周 晋, 路海明, 李衍达

(清华大学 自动化系 网络信息实验室,北京 100084)

Using Small-World to Devise Routing Algorithm for Unstructured Peer-to-Peer System

ZHOU Jin+, LU Hai-Ming, LI Yan-Da

(Web Information Laboratory, Department of Automation, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62778137, E-mail: zhoujin00@mails.tsinghua.edu.cn, http://webinfo.au.tsinghua.edu.cn/~zhoujin

Received 2003-03-25; Accepted 2003-10-08

Zhou J, Lu HM, Li YD. Using small-world to devise routing algorithm for unstructured peer-to-peer system. *Journal of Software*, 2004,15(6):915~923.
http://www.jos.org.cn/1000-9825/15/915.htm

Abstract: Peer-to-Peer systems have shown a great potential on file sharing in recent years, and developing efficient search techniques has become a crucial research problem. Most unstructured peer-to-peer systems with existing distributed routing algorithms can only run blind search, but not global search due to the lack of cache scheme. To address the problem, a novel key clustering algorithm is proposed, which divides the routing space into two layers, AUT layer and HUB layer. Such an algorithm can perform a well-ordered search from a global view. In order to improve scalability, theoretical results from small-world are utilized. In the improved algorithm, a few shortcuts with distant peers are inserted into the routing tables with some probabilities, and the average path length is reduced. The preliminary simulation results show that the key clustering algorithm with shortcuts is efficient and scalable.

Key words: P2P routing; clustering; small-world; unstructured peer-to-peer system

摘 要: 由于 peer-to-peer 系统在文件共享方面有着巨大的应用前景,peer-to-peer 搜索问题已成为目前学术界重点的研究问题之一.对于缺乏缓存机制的无组织 P2P 系统,已有的分布式路由算法缺乏全局导航能力,属于无序搜索.为此,提出一种 key clustering 算法,将路由空间分为 HUB 和 AUT 两层,从全局角度进行有序搜索.为提高 key clustering 算法的可扩展性,借鉴 Small-world 领域的研究成果,在路由表中以一定概率插入连接远距离节点的快捷连接,以缩短平均路径长度.初步仿真实验表明,引入快捷连接的 key clustering 算法具有良好的搜索能力和扩展性.

* Supported by the National Natural Science Foundation of China under Grant No.60003004 (国家自然科学基金)

作者简介: **周晋**(1977－),男,河北唐山人,博士生,主要研究领域为 P2P 搜索算法,合作信息检索;**路海明**(1973－),男,博士,助理研究员,主要研究领域为多代理系统,网络信息挖掘,分布式人工智能;**李衍达**(1936－),男,教授,博士生导师,中国科学院院士,主要研究领域为生物信息学,智能信息处理,信号处理.

关键词: P2P 路由;聚类;小世界;无组织 peer-to-peer 系统
中图法分类号:TP393 文献标识码: A

近来,peer-to-peer 系统(简称 P2P 系统)在文件共享和信息搜索等方面得到了越来越多的应用.P2P 系统由一组地位相等的节点构成,节点间可以直接通信,无须第三方参与.与集中式结构相比,P2P 结构在可扩展性、实时性、可靠性和负载平衡方面具有天生优势.

搜索算法是决定 P2P 系统性能的首要因素.Napster[1]采用了 Web 搜索引擎的集中式搜索,而不是真正的分布式搜索,但集中式搜索面临着信息量过载、拒绝服务攻击(denial of service)等一些难以解决的问题.

纯粹的分布式搜索算法在索引结构上可以分为两类:严格结构索引和自由结构索引[2].属于严格结构索引的典型算法有 Tapestry[3],Pastry[4],CAN[5]和 Chord[6],它们通过严格控制网络拓扑和文件存放位置,能有效地检索到结果,同时保证搜索步数在 $O(\log N)$的范围内(N 为节点总数),实现了可扩展性搜索,但搜索算法对节点的限制条件过多,需要严格控制网络拓扑和文件存储位置,所以,它们更适合运行于企业内网.

自由结构索引给予节点充分的自主性,仅根据搜索历史对索引记录进行适应性调整,因此应用前景广泛.该类方法的典型代表有 Gnutella[7]和 Freenet[8],其基本搜索原理分别为宽度优先(BFS)和深度优先(DFS).这两种算法鲁棒性虽强,但运行效率很低.扩散式的宽度优先搜索导致消息数量呈指数增加,回溯式的深度优先搜索导致耗时过长.本地索引(local index)算法[9]统计了邻居节点的信息,根据邻域信息选择合适方向进行搜索,以求减少消息量.但是它毕竟只利用了局部信息进行启发式搜索,缺乏全局视角,搜索性能改善程度仍然很有限.为了提高搜索性能,本文设计了一种键聚类(key clustering,简称 KC)算法,通过逐步调整将节点索引记录按照一定中心进行聚类,在物理网络层上建立具有全局视角的路由层.为了减少平均路径长度,我们利用小世界(small-world)的研究成果,将确定式聚类改为带有随机性的概率式聚类,在导航表中引入指向远距离节点的快捷连接(shortcut).在理论上,改进算法的平均路径长度的期望为 $O(\log 2N)$.

本文第 1 节给出了搜索问题的描述.第 2 节详细介绍了 key clustering 算法及其改进算法.在第 3 节中,通过仿真实验检验算法的性能.第 4 节给出结论与未来工作.

1 问题描述

本文讨论的 P2P 搜索算法将限定在无组织系统的范畴内.“无组织”的含义是指:(1) 网络拓扑结构自由;(2) 系统对节点无缓存要求,节点只需存储本地数据,不必为系统分担存储任务.

1.1 分布式搜索

在分布式搜索算法中,搜索命令(query)是沿节点连接进行传递的.根据一定策略选择消息的发送对象,邻居收到消息后,首先检索本地数据是否匹配,将匹配成功的消息沿原路径回传至初始节点;同时,选择合适的邻居继续扩散消息.搜索深度由 TTL(time-to-live)计数器控制,消息每前送(forward)一次,TTL 就减 1,当 TTL 减至 0 时,消息停止前送.一次 P2P 搜索的最终结果为所有的搜索分支的结果总和.

1.2 评价指标

综合前人的研究[9,10],我们用以下标准来评价搜索算法的性能:

- 搜索成功率(success rate)

$$\text{搜索成功率}=\frac{\text{搜索成功的数量}}{\text{搜索总数量}}.$$

- 平均路径长度(average path length)

平均路径长度是指搜索路径长度的平均值,平均路径长度是决定等待时间(latency)的直接因素.

- 消息数量(number of message)

消息数量会影响网络负载和节点的计算资源.在保证搜索成功率的前提下,减少消息数量是改善搜索性能

的有效手段.

2 Key clustering 算法和 Small-World 的应用

2.1 Key clustering算法

1) 索引构成

Key clustering 算法的索引采用了分布式哈希表(distributed Hashing tables,简称 DHT)[11],利用哈希函数将文件的属性(如文件名或描述性关键词)映射为键(key).将 key 按照升序排列,并首尾相接,形成环状空间,key 空间是位于物理节点层之上的上层结构(overlay),具有可搜索性.搜索时,query 也映射为 key.定义 key 之间的距离为它们在环状 key 空间上的最短距离,即 $D(a,b)=\mathrm{Min}\{|a-b|,|M-a-b|\}$,其中 M 为空间中 key 的总数.

每个节点拥有一个由中央服务器分配的聚类中心(center),节点根据搜索表现更新路由表内容,使内容向着 center 进行聚类.各个节点的 center 在 key 空间中均匀分布.为了保证系统能够及时追踪时刻变化的查询热点,中央服务器会根据近期 query 的分布为节点分配 center,以保证整个系统的索引均衡性.

节点在本地需要维护一张长度为 K 的路由表(见表 1).路由表分成导航表(HUB)和主题表(AUT)两部分,长度分别为 K_H 和 K_A($K_H+K_A=K$).AUT 表记录节点 ID 和代表其文件特征的 key,HUB 表记录节点 ID 及其 center.从功能上看,AUT 层面从全局角度按 key 的数值顺序重新编排文件记录,HUB 层面则按节点 center 的数值顺序排列节点,形成有序的路由层.系统运行时,二者分别按照一定的聚类策略进行更新.

Table 1 Routing table of KC algorithm

表 1 KC 算法的路由表

Node 003	center=4340	
	Key	Node ID
HUB	4 216	218
	4 855	82
	…	
AUT	4 217	264
	4 209	318
	4 532	403
	…	

2) 搜索过程

搜索时,首先检索本地文件,若与 query 匹配,则沿原路回送结果,同时更新路径上的节点路由表;否则,在 AUT 表中查找 query,若匹配,则前送消息至合适的节点;若 AUT 表不存在匹配节点,再从 HUB 表中寻找最接近 query 的节点(即 center 与 query 接近),并发送 query 至该节点.

3) 路由表更新策略

设搜索路径的长度为 L,搜索命令为 q,从起始节点到目标节点依次是 $n_1,n_2,\ldots,n_L$.由于路径上的节点参与了对 q 的搜索,故可以认为它们与 q 的相关性较大,因此,在搜索成功后,KC 算法会对这些节点的路由表进行更新操作.

• HUB 表更新策略

对于 $n_x(x=1,2,\ldots,L-2)$,进行 HUB 表的更新:

① 令等待加入的新记录 $key_{Ins}=center(n_{L-1})$,$center(n_{L-1})$即为 n_{L-1} 的聚类中心,相应地记录为$\langle key_{Ins},n_{L-1}\rangle$.之所以选择 n_{L-1} 为学习目标,是因为 n_{L-1} 作为路径上最后一个消息传递者,为搜索成功做出了最重要的贡献;

② 若 HUB 表未满,则加入新记录$\langle key_{Ins},n_{L-1}\rangle$,更新结束;若 HUB 表已满,则执行③;

③ 待删除的旧记录 key_{Del} 为 n_x 的 HUB 表中距离 $center(n_x)$最大的记录 $key_{Del}=\mathrm{Max}\{D(key_x,center(n_x))\}$,$key_x$ 是 n_x 的 HUB 表中的任意一条记录;

④ 如果 $D(key_{Ins},center(n_x))<D(key_{Del},center(n_x))$,则用新记录取代旧记录;否则,不做任何改动.

• AUT 表更新策略

对于 $n_x(x=1,2,\ldots,L-1)$,进行 AUT 表的更新:

① 因为在 n_L 处搜索到 q,所以 n_L 的文件中必包含 q,路径上的节点将记录该信息.令等待加入的新记录 $key_{Ins}=q$,相应地记录为$\langle key_{Ins},n_L\rangle$,;

② 若 AUT 表未满,则加入新记录$\langle key_{Ins},n_{L-1}\rangle$,更新结束;若 AUT 表已满,则执行③;

③ 待删除的旧记录 key_{Del} 为 n_x 的 AUT 表中距离 $center(n_x)$最大的 key,即 $key_{Del}=\mathrm{Max}\{D(key_x,center(n_x))\}$,$key_x$ 是 n_x 的 AUT 表中的任意一条记录;

④ 如果 $D(key_{Ins},center(n_x))<D(key_{Del},center(n_x))$,则用新记录取代旧记录;否则,不做任何改动.

我们希望经过足够多次的更新后,HUB 和 AUT 中的 key 可以有效地聚类于 center.

4) 算法原理解释

直观上讲,key clustering 想法就是让各个节点在整个 key 空间上各司其职,分工合作.各司其职通过 AUT 来实现,AUT 负责管理 key 空间内一段区域的 key(非强制性),区域中点即为 center.AUT 表的聚类可以在原来节点层面之上重建全局有序的管理层;HUB 记录节点的 center,center 代表了节点的职能,通过 HUB,节点能够了解彼此的职能,聚类后的 HUB 可以在 AUT 层面上建立有序的路由层.

在图 1 中,纵向代表节点的不同层面,横向代表不同的节点.KC 算法在原有节点层上建立了 AUT 层和 HUB 层,节点按照 key 的数值顺序排列.节点层指向 AUT 层的箭头表示原有节点描述本地文件的 key 信息在 AUT 层如何进行重置,AUT 层指向 HUB 层的箭头表示 AUT 层的描述节点职能的 center 信息在 HUB 层如何进行重排.

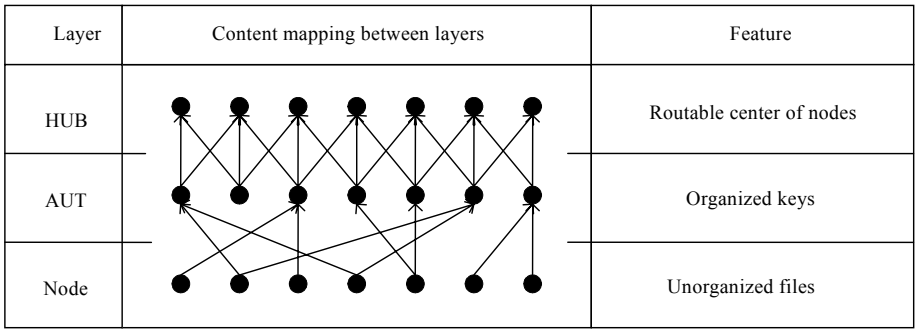


Fig.1 Content mapping between layers of KC algorithm

图 1 KC 算法层次间的映射关系

在节点层,key 的分布无任何规律可循,在该层面无法实现有效的搜索;AUT 层对 key 的分布重新进行了安排,key 分别向所属节点聚类,即每个节点的 AUT 分管 center 左右一定范围内的 key;在 AUT 层之上,HUB 层又建立了路由层,负责建立节点间的联系,即每个节点的 HUB 分管本节点 center 左右一定范围的其他节点 center,这样,每个节点都可以了解到左右邻居的情况(center 相邻或相近),从而保证搜索得以有效进行.具体地,在 key 空间上,搜索按 key 递增或递减方向前进.

AUT 记录了其他节点文件的 key,若 query 与 AUT 的某条记录匹配,说明距离目标节点仅一步之遥;HUB 确保了搜索沿着正确方向前进,而不必走重复路.我们可以形象地认为,AUT 存储了文件(事实上距文件尚有一步),HUB 存储了导航信息.KC 算法的路由表十分类似于 WWW 上的情形,许多网页既含有信息文件(如,文本、图片),又含有指向主题相近网站的链接,也就是说,网页都有权威性(authority)和中心性(hub)两种特性.将路由表分拆成 AUT 和 HUB 的想法也正源于此.

HUB 与 AUT 拥有相同的聚类中心的原因是,维护导航方向和搜索目标的一致,否则导航失去意义.HUB 与 AUT 的关系是相互协作,缺一不可的,HUB 将各个节点串连起来,建立可搜索结构,而最终命中目标要通过 AUT 来实现.

2.2 应用Small-World改进KC算法

1) Small-World 及其应用

著名的 Stanley Milgram 实验发现,通过平均 6 人次的熟人传递就可以把社会中任意两个人联系起来,这种现象称为 Small-World 现象.Milgram 实验揭示了两个发现:1) 短链效应普遍存在;2) 人们可以找到短链.第 2 个发现说明,当网络呈现某种拓扑结构时,仅利用局部信息就可以有效地找到短链.这个发现为分布式信息搜索提供了契机.

Watt 和 Strogatz 提出的模型(简称 WS 模型)[12]是一种常用的 Small-World 模型:N 个节点分布在一个圆环上,初始状态时,每个节点有 k 个连接,分别连向最近的 k 个点.然后,依次调整各节点的连接,以概率 p 随机地改变连接的终端,但避免连向节点本身.

记 $D(i,j)$为节点 i 和 j 之间的最短距离,平均路径距离 L 的计算公式如下:

$$L=\frac{1}{n(n-1)/2}\sum_{1\le i\ j\le n}D(i,j)\ .$$

当 $p\approx 0$ 时, $L\sim\frac{n}{2k}$,此时网络拓扑呈规则状态;当 $0.001<p<0.01$ 时, $L\sim\frac{\ln n}{\ln k}$,此时节点不仅与相邻节点存在连接,还与远距离节点建立了少数的 shortcut,正是这些 shortcut 有效缩短了 L,使整个网络呈现出 small-world 特征.

在 KC 算法中,节点按 center 的数值顺序依次放置在环状 key 空间上,搜索消息沿圆环依次传递,平均路径长度为 $O(N)$.这导致平均路径长度随着节点总数的增长成比例地增长,搜索成功率随之下降,这严重影响了 KC 算法的实用性.

为了解决该问题,我们借鉴 Kleinberg 的研究成果[13]改进 KC 算法.Kleinberg 模型是一个 k 维的格网络(lattice network),对于节点 u 发出连接,以正比于 $[d(u,v)]^{-r}$ 的概率与 v 建立连接(其中 v 为任意非 u 节点;$d(u,v)$表示 u,v 的网格距离;r 是常数,叫做聚类指数,用以控制节点的聚类程度).简单来说,一个节点与近邻节点建立连接的可能性较大,与远方节点建立连接的可能性较小.

由 Kleinberg 证明得知,对于 k 维格网络,当且仅当 $r=k$ 时,存在一种分布式搜索算法,使得平均路径长度是 $\log N$ 的多项式规模.特别地,对于 1 维格网络,当 $r=1$ 时,L 是 $\log N$ 的多项式规模.KC 链表属于 1 维的格网络,当以 $d(u,v)^{-1}$ 的概率建立连接时,平均路径长度的期望值为 $O(\log^2 N)$.

2) Enhanced key clustering 算法

基于 Kleinberg 的研究成果,我们对 KC 算法的 HUB 表更新策略进行了改进,提出 EKC(enhanced key clustering)算法,而 AUT 表更新策略仍与 KC 算法相同.

EKC 算法的 HUB 表更新策略如下:

对 $n_x(x=1,2,\ldots,L-2)$进行 HUB 表的更新:

① 令等待加入的新记录 $key_{Ins}=center(n_{L-1}),center(n_{L-1})$即为 n_{L-1} 的聚类中心,相应的记录为$\langle key_{Ins},n_{L-1}\rangle$;

② 若 HUB 表未满,则加入新记录$\langle key_{Ins},n_{L-1}\rangle$,更新结束;若 HUB 表已满,则执行③;

③ 计算 HUB 表中每条记录的删除概率,记 $u=center(n_x)$,则 $key\ v$ 的删除概率为

$$P(v)=\frac{D(u,v)^{-1}}{\sum_{w}D(u,w)^{-1}}\ ,$$

其中 w 为 HUB 表中的任意 key.根据删除概率随机选择一个 key 为待删除记录,记为 key_{Del};

④ 重新计算 key_{Del} 和 key_{Ins} 的删除概率,

$$P(key_{Del})=\frac{D(u,key_{Del})^{-1}}{D(u,key_{Del})^{-1}+D(u,key_{Ins})^{-1}}\ ,$$

$$P(key_{Ins})=\frac{D(u,key_{Ins})^{-1}}{D(u,key_{Del})^{-1}+D(u,key_{Ins})^{-1}}\ .$$

根据删除概率随机确定删除对象,对 HUB 表作出相应调整.

EKC 算法按照概率更新 HUB 表,可以为 HUB 表随机引入少数的 shortcut.

3 仿真实验

3.1 实验设置

在仿真实验中,假设各个节点有相同数量的邻居,信息资源在系统中均匀分布.在仿真中,系统由N个节点构成.每个节点从一个容量为 10 000 的 key 库中随机选择 KPN 个 key 来代表本地文件特征.每个节点随机选择 NPN 个节点作为邻居.每次实验进行多次仿真,每次仿真进行多次搜索.每次搜索时,随机选定 query 发起节点,随机选定 query.在搜索成功后更新搜索路径节点的路由表.其他参数说明见表 2.

Table 2 Definition and value of symbols

表 2 仿真实验的符号说明与参数设定

Description	Symbol	Value
Total number of nodes	N	500
Numbers of keys in the pool	KPL	10 000
Number of keys per node	KPN	3
Number of neighbors per node	NPN	3
Size of routing table	RTS	100
Size of HUB table	HUBS	25
Size of AUT table	AUTS	75
Times of searches	Sear	100 000
Times of simulations	Simu	20
Time to live	TTL	7

3.2 3种算法的比较

为了考察 KC 和 EKC 算法的搜索性能,我们选择 Local Index 为比较对象.三者的路由表总容量均为 100,KC,EKC 的 HUB 和 AUT 容量分别为 25 和 75.

表 3 比较了 3 种算法的搜索性能,数据为 20 次仿真实验的平均值,每次实验搜索次数为 100 000 次,其他参数取值见表 2.可以看出,在分层和聚类的作用下,路由表性能得到了明显的提高.KC 和 EKC 的成功率比 Local Index 提高了 1 倍多,而在平均消息数量和平均路径长度上,二者明显少于 Local Index.

Table 3 Performance of three algorithms

表 3 3 种算法的搜索性能

Policy	Success rate (%)	Average message number	Average path length
Local index	21.52	108.17	6.56
Key clustering	48.87	70.66	5.74
Enhanced key clustering	55.34	50.19	3.61

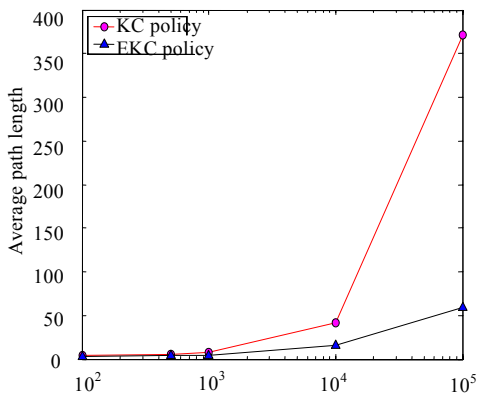


Fig.2 Average path length to different node number

图 2 不同节点数量下的平均路径长度

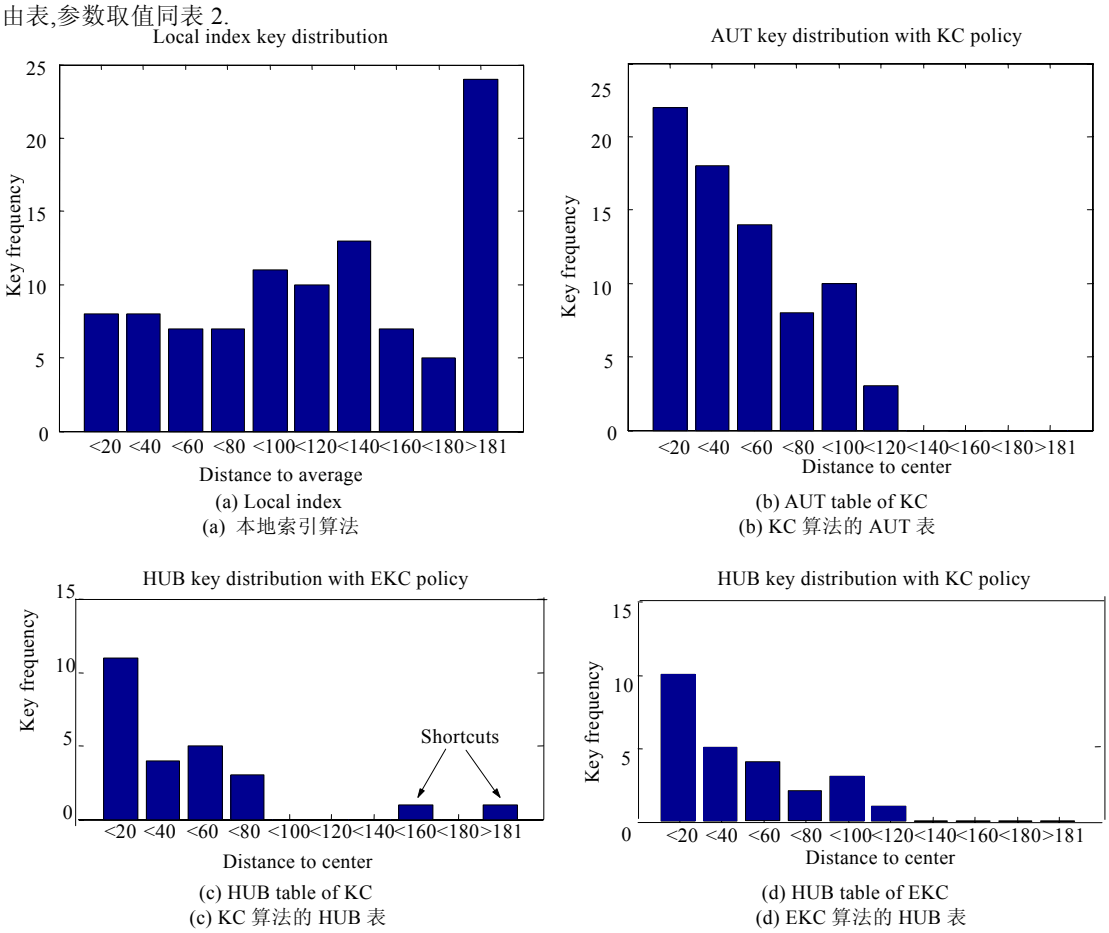
与 KC 相比,EKC 在性能上有进一步的提高.由于 EKC 的 HUB 存在少量的 shortcut,加快了搜索进程,使之平均消息数量和平均路径长度比 KC 分别减少了 29.0%和 37.1%,从而为成功率带来了 13.2%的提升.

图 2 比较了 KC 和 EKC 的在不同节点数量下的平均路径长度变化,除节点总数和 TTL 外,实验参数设定同表 2,从图中可以更清楚地看出,随着节点数量的增加,KC 曲线的抬升速度大大快于 EKC 曲线,与理论上的指数趋势相符.由于受节点数量增长的影响小,EKC 比 KC 具有更好的推广性.在未来的工作中,我们将对节点数量与平均路径长度给出更精确的定性分析.

3.3 分析与讨论

1) Key 分布

为了直观地理解 key clustering 思想,图 3 给出了各种算法的路由表的 key 分布.图 3(a)中横轴代表 key 与平均值的距离,图 3(b)~图 3(d)中的横轴代表 key 与 center 的距离.图 3 中横坐标代表区间段,例如,“<80”表示[61,80]区间段,“>181”表示[181,∞]区间段,纵坐标为落在该段的 key 的个数.图 3 把 key 与 center(或平均值)的距离划分为若干区间段,统计每段的 key 数量,以考察算法对 key 分布的影响.图示数据选自一次实验的某节点的路由表,参数取值同表 2.



(a) Local index
(a) 本地索引算法

(b) AUT table of KC
(b) KC 算法的 AUT 表

(c) HUB table of KC
(c) KC 算法的 HUB 表

(d) HUB table of EKC
(d) EKC 算法的 HUB 表

Fig.3 Distribution of keys under different policies

图 3 不同路由表的 key 分布

Local index 的数据记录的是附近邻居的 key,它们之间不存在明确关系,分布应属于均匀分布.应用了聚类更新策略的 KC 和 EKC(包括 HUB 与 AUT)在 key 分布上呈现出明显的聚类特征(EKC 与 KC 的 AUT 调整算法相同,key 分布近似,故在图 3 中仅给出了 KC 的 AUT 分布).

需要注意的是,EKC 的 HUB 调整算法在考虑与 center 距离的基础上加入随机性因素,以一定的概率决定新记录是否加入,这不同于 KC 的严格聚类做法.因此,EKC 的 HUB 中存在少数非聚类 key.事实上,聚类 key 起到了近邻连接的作用,非聚类 key 则相当于 shortcut.

2) HUB 层搜索

从图 4 中我们可以直观地看出 KC 搜索和 EKC 搜索的区别.如图 4(a)所示,在 KC 算法的 HUB 层中,连接规则分布,从 A 到 B 需要通过邻居依次传递消息,路径长度正比于 D(center(A),center(B));而 EKC 算法中存在着少数的随机 shortcut,shortcut 可以起到缩短路径的作用,在图 4(b)中,A 到 B 的最短路径长度仅需 3 步.

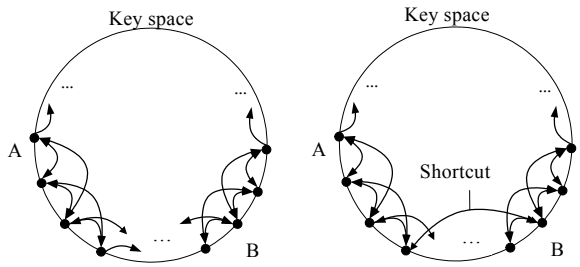


Fig.4 An example of topology in HUB layer

图 4 HUB 层拓扑示意图

需要指出的是,在搜索起步阶段,HUB 层可以准确地指引搜索方向,但当足够接近 query 时,HUB 层的导航作用便不再明显,因为路由表(包括 AUT 和 HUB)的 key 在距离上与 center 并不是严格意义上的最近.举个例子,与节点 v 相比,节点 u 的 center 与 query 更接近,但可能 u 的 AUT 不能满足 query,而 v 可以.因此,在搜索足够接近 query 后,可能在 center 相近的几个节点间跳跃,这类似于遍历搜索,会对路径长度有一定的影响.但从整体来看,HUB 层对搜索的全局导航能力是有效的.实验已经验证了这一点.

3) HUB 表和 AUT 表的容量比例

路由表的总容量固定,但 HUB 表和 AUT 表的容量比例并没有限制,从功能上看,二者分别起到了导航和管理 key 的作用,缺一不可.图 5 给出了 HUB 表容量大小与成功率的关系,除了 HUB,AUT 的容量以外,实验取值同表 2.随着 HUB 容量占路由表总容量的比例增大,成功率曲线表现出先升后降的特征,在大约 0.25 处,达到极值,此时 HUB 容量 25,AUT 容量 75.如果 HUB 表比例过低,搜索的导航信息不足,则成功率会严重下降;如果 HUB 表比例过高,AUT 表相对变小,所管理的 key 数目过少,将直接影响搜索成功率.

目前,关于 HUB 比例的优化问题缺乏理论分析,只能通过实验手段确定最优值.该优化问题是今后研究的问题之一,涉及因素可能包括:节点总数、key 总数以及 key 的序号分布等.

4) 容错性(fault tolerance)

节点的自由退出是无组织系统必须要考虑的问题.图 6 显示了节点退出对 EKC 算法的成功率的影响,以所有节点在线的成功率为基准,纵轴代表当前成功率与基准的比值.实验中,要保证系统中存在 3 份以上的满足 query 的资源,以保证在部分节点退出时,搜索仍可能找到结果,其他参数设定同表 2.

图 6 中,随着节点退出比例的增大,曲线开始下降缓慢,当退出率=35%时,成功率仍能保持在 80%以上.之后,曲线才加速下降,当退出率=50%时,成功率仅为原来的 10%左右.

在一定的退出率下,除了退出节点的路由表不再发挥作用,工作节点的路由表中涉及到退出节点的记录也失效,图 6 说明了尚存的少数 HUB 连接仍能有效地指引搜索方向,同时,聚类的 AUT 表缩小了目标查找范围,节省了消息发送数量.可见,EKC 具有良好的容错性.相比之下,组织无序的路由表的搜索能力完全依赖于有效记录的数量,在节点退出时,搜索的盲目性会更加突出,成功率往往会随着退出率成比例下降.

4 结 论

本文针对无组织 P2P 系统的分布式路由算法搜索无序问题进行研究,提出了 Key clustering 算法.它将路由空间分 HUB 和 AUT 两层,从全局角度构建路由表,从而实现有序搜索.为了改善平均路径长度的期望等于 $O(N)$ 的状况,我们借鉴小世界的研究成果,在路由表中以一定概率插入连接远距离节点的快捷连接.在理论上,平均路径长度的期望值改善为 $O(\log^2 N)$,提高了算法的可扩展性.初步仿真实验表明,引入快捷连接的 Key clustering 算法具有良好的搜索能力、可扩展性和容错能力.未来的研究问题包括:1) HUB 表与 AUT 表的容量配比问题;2) 在横向的链式结构上,如何建立纵向的层次模型;3) 对 AUT 表进行层次聚类,以求进一步减小路由表,提高搜索

效率.

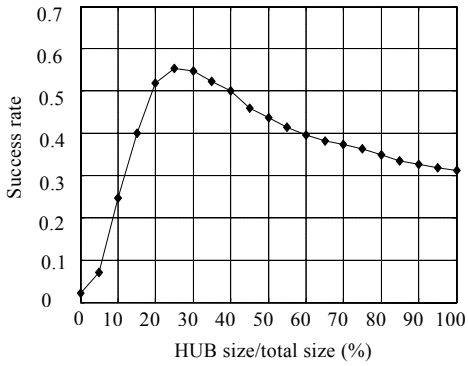


Fig.5 Success rate vs. HUB size
图 5 成功率与 HUB 表容量的关系

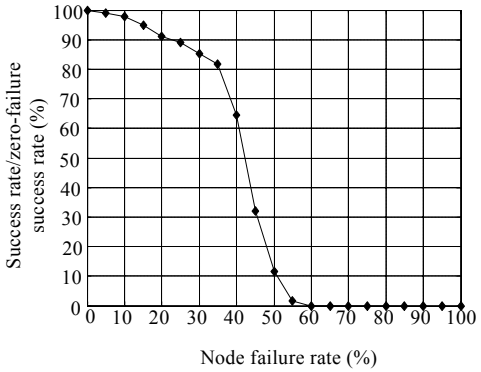


Fig.6 Success rate vs. node failure rate
图 6 成功率与节点退出率的关系

References:

[1] Napster. 2003. http://www.napster.com

[2] Daswani N, Garcia-Molina H, Yang B. Open problems in data-sharing peer-to-peer systems. In: Calvanese D, Lenzerini M, Motwani R, eds. Proc. of the 9th Int'l Conf. on Database Theory (ICDT). Heidelberg: Springer-Verlag, 2003. 1~15.

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

[4] Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Guerraoui R, ed. Proc. of the Middleware 2001. Heidelberg: Springer-Verlag, 2001. 329~350.

[5] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Govindan R, ed. Proc. of the ACM SIGCOMM 2001. ACM Press, 2001. 161~172.

[6] Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for Internet applications. In: Govindan, ed. Proc. of the ACM SIGCOMM 2001. ACM Press, 2001. 149~160.

[7] Gnutella. 2003. http://gnutella.wego.com

[8] Clarke I, Sandberg O, Wiley B, Hong TW. Freenet: A distributed anonymous information storage and retrieval system. In: Faderrath H, ed. Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability. Berkeley: Int'l Computer Science Institute, 2000. 43~58.

[9] Yang B, Garcia-Molina H. Improving search in peer-to-peer networks. In: Sivilotti PAG, ed. Proc. of the Int'l Conf. on Distributed Computing Systems. IEEE Computer Society, 2002. 5~14.

[10] Balakrishnan H, Kaashoek MF, Karger D, Morris R, Stoica I. Looking up data in p2p systems. Communications of the ACM, 2003, 46(2):43~48.

[11] Watts DJ, Strogatz SH. Collective dynamics of small-world networks. Nature, 1998,393:440~442.

[12] Kleinberg J. The small-world phenomenon: an algorithmic perspective. Cornell Computer Science Technical Report, 99-1776, 2000.