ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.3, March 2007, pp.714−721
DOI: 10.1360/jos180714

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于分布式哈希表的混合对等发现算法*

杨 峰[1+], 李凤霞[2], 余宏亮[1], 战守义[2], 郑纬民[1]

[1](清华大学 计算机科学技术系 高性能计算研究所,北京 100084)

[2](北京理工大学 计算机科学技术学院,北京 100081)

A Hybrid Peer-to-Peer Lookup Service Algorithm on Distributed Hash Table

YANG Feng[1+], LI Feng-Xia[2], YU Hong-Liang[1], ZHAN Shou-Yi[2], ZHENG Wei-Min[1]

[1](Institute of High Performance Computing, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

+ Corresponding author: Phn: +86-10-62774912, Fax: +86-10-62774912, E-mail: yangfeng@tsinghua.edu.cn

Yang F, Li FX, Yu HL, Zhan SY, Zheng WM. A hybrid peer-to-peer lookup service algorithm on distributed hash table. *Journal of Software*, 2007,18(3):714−721. http://www.jos.org.cn/1000-9825/18/714.htm

Abstract: An application using a distributed hash table (DHT) with N nodes must choose a DHT protocol from the spectrum between $O(1)$ lookup protocols and $O(\log N)$. However, various applications under different network churns require that an idea DHT would be adaptive in according with the churn rates. ROAD (routing on active and demand), a new lookup algorithm, adjusts itself to provide the best performance across a range of lookup delay and churn rates. The key challenges in the design of ROAD are the algorithms that construct the routing table's size and decrease the delay. It will speed up the lookup process and reduce the service delay with the expressed routing table and power sorting multicast algorithm. Simulations show that ROAD maintains an efficient lookup delay versus churn rate tradeoff than the existing DHTs. ROAD should be expanded into a mechanism that provides some kinds of lookup services with a range of qualities of service through super-peers choosing methods.

Key words: lookup delay; churn rate; hybrid route; multicast; distributed hash table

摘 要: 使用分布式哈希表(distributed hash table,简称 DHT)的应用系统必须在 $O(1)$发现算法和 $O(\log N)$发现算法系列中选择适应的 DHT 协议.但是,不同网络波动程度的应用场景要求理想的 DHT 协议根据网络波动率能够自适应地调整.提出一种发现算法 ROAD(routing on active and demand),在延时和波动率之间自适应地调整以提供更好的性能.设计 ROAD 的关键挑战是构建路由表和降低延时的算法.通过构建加速路由表,加快发现服务的速度,降低消息转发的延时,并通过幂次序组播算法改善对超级点的依赖性.模拟实验显示,与现有 DHT 算法相比,ROAD 维护了一种高效发现延时与波动率的折衷.选择不同质量类型的超级点,ROAD 可以扩展成满足不同服务需要的发现机制.

关键词: 发现延时;波动率;混合路由;组播;分布式哈希表

* Supported by the National Natural Science Foundation of China under Grant Nos.60433040, 60603070 (国家自然科学基金); the China Postdoctoral Science Foundation under Grant No.2005038064 (中国博士后基金)

Received 2004-03-24; Accepted 2006-04-21

中图法分类号: TP393 文献标识码: A

基于 DHT(distributed hash table)的分布式资源组织、管理与发现服务正在应用到网格、Web 服务[1]、对等计算等分布式环境中,成为构建大规模分布式应用的基础结构.然而,不同的网络场景下节点和网络连接的动态性不同,要求资源组织、管理与发现服务能够适应系统的不同变化程度,保证部分节点或网络失效时服务的可用性[2].

研究[3]发现,不同的物理网络波动情况差别很大.研究[3]在 60 小时内监控了 Gnutella 网络中 17 000 个节点,显示 Gnutella 的平均生存时间是 2.9 小时,等于说 10 万个节点的系统每秒钟大约 19 个成员关系变化;并在一个月内监控了 65 000 个 Microsoft 企业网中的节点,显示企业网中 54 610 个节点中半数以上在 95%的时间内可以存活.而移动 Ad hoc 网络则相对变化较快[4],尤其是未来 P2P 技术重要应用场景——军事战术通信系统的变化程度很难确定.

现有发现服务都有其自身特有的网络应用场景.它在某一种场景下运行性能良好,可能会在另一种场景中性能下降.影响发现服务的因素包括:网络的波动(包括节点的加入、退出、失败、迁移、并发加入过程、网络分割等)、网络延迟、传输带宽和对等点的计算能力等[2].还没有一种方法能够适应不同的网络环境和动态变化程度.

我们提出一种混合型 P2P 发现服务算法,在同一个 DHT 内实现了不同路由策略,可以根据网络波动程度自适应地选择最合适的策略进行路由.给出了一种加速路由表的构造算法,每个节点路由时尽量选择超级点构成路由路径以此来获得高效率的发现服务质量.为了降低层次式结构对超级点(super peer)的依赖性,我们给出一种幂次序组播算法,降低了超级点的维护代价.

本文第 1 节分析两类对等发现方法的性能.第 2 节详细介绍混合发现服务设计,其中第 2.2 节给出加速路由表的构造算法和基于 QoS 度量尺度的混合路由策略,第 2.3 节介绍幂次序组播算法.最后是模拟测试结果和结论.

1 网络波动对 P2P 发现算法的影响分析

结构化 P2P 系统大都采用 DHT 进行资源组织、管理与发现,如 CAN,Chord[5],Pastry,Tapestry,Koorde 和 Viceroy 采用的多跳路由算法进行发现服务,即路由表仅需要维护 $O(\log N)$或 $O(1)$个节点状态,发现服务仅需要 $O(\log N)$跳路由.某一个节点加入或退出网络时,最少需要 $O(\log N)$或 $O(\log^2 N)$条消息维护路由表的一致性.每个节点维护有限的成员关系信息,有效地解决较大网络波动问题,但却以增加延时为代价.由于它不必维护去往所有节点的路由,仅在没有去往目的节点路由的时候才按需进行路由发现,我们称其为按需发现服务.

另外一种是 Kelips[6],Twins[7],One-hop[8]采用基于超级点结构的单跳路由算法的发现服务,每个超级点路由表维护全局路由状态,路由表开销为 $O(N)$或 $O(\sqrt{N})$;每个发现服务仅需要 $O(1)$跳;发现速度快,延时小,但维护代价高,受网络波动的影响大.我们称其为主动发现服务.

我们的目的是评价这些工作在网络不同状态下的性能,为本工作选择路由策略提供依据.我们定义两个评估参数:

① 查找失败率 Q:即发起路由失败的概率,定义 Q 为查找失败率.假设某次路由经过 n 个路由节点到达目的点,每个节点失败的概率为 p,则 $Q=1-(1-p)^n$.

② 相对延时:它应该使得路由的平均 RDP (relative delay penalty),即 DHT 路由延时与底层网络最短延时之间的比值尽可能地小.由于 $RDP=\sum_{hop} t_{(i,j)} / t_n$,其中 t_n 为发起请求的源节点到目的节点之间的底层网络延时,$t_{(i,j)}$为每一跳选择的两个中间路由节点的底层网络延时,hop 为跳数.由于 $t_{(i,j)}$和 t_n 依赖具体的网络场景,我们使用 hop 来描述路由延时.

另外,定义网络波动率为 r(1/秒),表示每秒钟加入或退出网络的节点个数.由于最差网络波动模型为并发加入或退出模型,即在事件发生后 t_r 秒内,关于这些事件的所有路由记录都不正确,失败分布函数遵循平均分布,每

节点失败的概率为[8,9]

$$p=\frac{r*t_r}{N} \tag{1}$$

其中:t_r为路由表维护周期;N为P2P网络的节点总数.由于研究[10]证明Chord在所有按需发现服务中对网络波动表现出最好的性能,因此,我们选择Chord算法代表按需路由与OneHop[9]路由算法进行比较.主动路由的查找失败率按照式(1)计算.按需路由的查找失败率按照式(2)计算:

$$E(Q)=\frac{1}{N}\sum_{n=1}^{N-1}\sum_{i=1}^{i_m}(1-p)p^{i_m-i}P(n-2^i) \tag{2}$$

由于主动路由的跳数为常数,因此不再给出主动路由的相对延时.Chord的相对延时计算公式为[10]

$$E(hop)=\frac{1}{N}\sum_{n=1}^{N-1}\sum_{i=1}^{i_m}h(n) \tag{3}$$

我们定义网络波动发生在Chord执行稳定算法之后,此时邻居的状态始终正确,这样,$P(0)=P(1)=0$.结果见表1,其中,网络规模定为2^{10}个点,t_r定为30秒.

Table 1 Lookup failure rate and delay of different churn rates

表1 不同波动率下发现服务的查找失败率和延时

Nodes num per second/ Nodes num per 30 seconds	Failure rate		Latency (hops)
	Lookup on active	Lookup on demand	
0.1/3	0.003	0	5.25
2/60	0.06	0	5.25
4/120	0.12	0	5.87
8/240	0.24	0	6.72
16/480	0.48	0	8.07

由表1可以看出:主动路由在变化速度慢的网络环境下等待延时最小,查找速度快.由于$t_r=Q\times N/r$,随着网络波动的加快,主动路由为了保持查找失败率,必须减少路由表维护周期.这样,一方面维护开销加剧,性能下降;另一方面,路由表的更新速度也无法跟上网络波动速度.而按需路由只要维护正确的邻居关系,就可以保持查找失败率不变或平稳下降,但却以增大延时为代价.研究[4]也发现:当Chord算法应用在DNS服务时,最差需要20跳才能解析DNS.但式(3)也证明:导致*hop*值增大的主要因素是路由节点的失败概率,只要选择可靠性高的路由节点就可以减少*hop*值.

综上所述,当网络波动速度处于静止或缓慢时,我们选择主动发现服务的路由策略提供快速而准确的查找,获得较小的延时;网络波动增大,我们选择按需发现服务通过选择多条路由路径保持查找率不变或平稳下降;当网络波动进入快速剧烈时,仅有洪泛算法可以适应这种变化.

2 ROAD设计

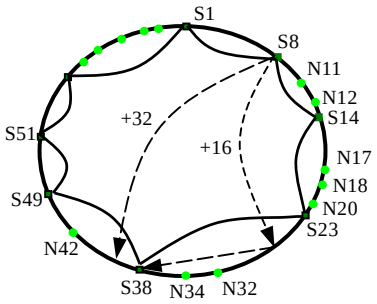
根据上述分析,我们选择主动路由和按需路由构成一个新的混合路由算法——ROAD(routing on active and demand).我们的目的主要是,在不同的波动情况下,保持路由的效率,同时使得请求的路由路径尽量短,以此来获得系统对波动程度高效率的反应.

2.1 ROAD结构

如图1所示,ROAD结构主要包括两部分:主动路由和按需路由.对于N个节点的P2P网络,通过DHT算法所有节点分布在一个环上,我们称其为外环,DHT算法为每个节点分配一个ID.环采用Chord路由算法.同时选择M个超级点构成一个内环,内环采用主动路由算法.所有点采用同一个DHT算法分配ID.

假设第i个超级点ID为s_i,第i+1个超级点ID为s_{i+1},s_{i+1}负责管理ID空间(s_i,s_{i+1})内的所有普通节点,这些普通节点和s_{i+1}构成一个弧(如S23,N20,N15),其中s_i为s_{i+1}的邻居,s_i通过心跳协议维护s_{i+1}的状态.令I为弧数目,G_i为第i弧的所有节点;g_i为第i弧.ROAD可以定义为无向图(G,U),其中:$G=\{g_1,g_2,\ldots,g_I\}$;U为$G=\{g_1,g_2,\ldots,g_I\}$中任意2个弧的边.由于超级点之间采用主动路由,(G,U)被定义为连通的,即任意(g_i,g_j)都有一个有向路径

$(g_i,g_j)\in U$.令 $S_i\subseteq G_i$ 为第 i 弧的超级点集合.若对所有 $i=1,\ldots,I,S_i=G_i$ 即所有点都是超级点,此时,ROAD 退化成主动路由结构.若对所有 $i=1,\ldots,I,S_i=\varnothing$,即没有超级点,ROAD 又退化成按需路由结构.



Finger of S14

S14+1	N17
S14+2	N17
S14+4	N18
S14+8	S23
S14+16	S38
S14+32	S49

Finger of S8

S8+1	N11
S8+2	N11
S8+4	S14
S8+8	S23
S8+16	S38
S8+32	S49

Super-Peer table of S23

ID	Ip address	T (ms)
S1	IP1	23
S8	IP8	125
S14	IP14	62
S23	IP23	17
…	…	

Indexes of S23

ID	IP address
N17	IP17
N18	IP18
N20	IP20
…	…

Fig.1 Construction of ROAD and its express routing tables

图 1 ROAD 结构和加速路由表

2.2 ROAD路由

资源路由和定位的效率决定了发现服务的质量.在请求转发过程中,路由表如何提供一个接近自己的可靠节点,使得请求能够可靠、快速地转发下去,是决定路由与定位效率的关键因素.相对于普通节点,超级点具有更高的可靠性和更低的延时,我们的目的就是尽量选择超级点填充路由表.因此,把经过超级点填充的路由表称为加速路由表.

2.2.1 加速路由表构造算法

在路由信息的处理方式上,我们继承了 Chord 对路由信息的划分,即前驱(predecessor)、后继列表(successor list)和 finger 表.其中,把 finger 表改造成了加速路由表(Qfinger).令节点 ID 号的长度为 m 位,每个节点的 Qfinger 表最多包含 m 条记录,Qfinger 算法如下:

定义 **1**. $g[i]$为 Qfinger 表中第 i 条记录 $finger[i]$选择路由点的 ID 空间,$1\le i\le m$,则

$$g[i]=[(n+2^{i-1}) \bmod 2^m,(n+2^i) \bmod 2^m].$$

定义 **2**. S_i 为 ID 空间 $g[i]$内所有超级点 ID 的集合,G_i 为 ID 空间$[(n+2^{i-1}) \bmod 2^m,n-1]$内所有点 ID 的集合.

若 $S_i\ne\varnothing$,第 i 条记录 $finger[i].node=\min\{s_k|s_k\in S_i,k=1,2,\ldots,|S_i|\}$.

若 $S_i=\varnothing$,第 i 条记录 $finger[i].node=\min\{g_k|g_k\in G_i,k=1,2,\ldots,|G_i|\}$.

以图 1 超级点 S8 构建 finger 表为例,S8 构建 $finger[1]$时,在 ID 空间[9,10]之间查找.未发现任何节点,继续向前(顺时针)查找,直到发现第 1 个节点 N11,$finger[1].node$=N11.构建 $finger[5]$时,在 ID 空间[24,40]之间查找.发现第 1 个节点 N32,$finger[5].node$=N32.由于 N32 为普通点且未到达空间边界,继续向前查找,发现第 1 个超级点 S38,$finger[5].node$=S38.这样,Qfinger 可能跨越多个普通点.如 S8 在构建 $finger[4]$时,跨越了 N17,N18,N20 点.但是,这 3 个点的信息出现在 S14 的 finger 表和 S23 的索引表内.也就是说,ROAD 增大了 finger 的距离,但没有减少路由信息,同时保证了路由的可靠性.

除了需要构建 Qfinger 表以外,超级点还要维护超级点表和负责的弧内普通点的索引表.一个超级点表和索引表如图 1 所示.

2.2.2 混合路由策略

如何在多种路由策略之间进行自适应的、平滑的切换是混合路由的关键[1].ROAD 采用了一种自动的切换机制:当超级点表出现如下两种状态之一时,可以自动切换到按需路由策略.① 没有去往目的节点的超级点,可能是由于节点加入信息还没有及时扩散;② 访问负责目的节点的超级点失败,可能是由于网络波动速度加快导致查找失败.

大部分发现服务的路由算法都通过设定等待周期来判断超级点访问失败.然而,由于不同的网络应用场景,很难找到一个通用的等待周期.我们借鉴工作[4]中的 QoS 度量方法自适应地计算等待周期.

定义 **3**. $kt_{(i,j)}$定义为节点 i 第 k 次访问超级点 j 的延时,单位 ms.

定义 **4**. $t^k_{(i,j)}$ 定义为节点 i 第 k 次访问超级点 j 结束后的等待周期,单位 ms.

$$t^k_{(i,j)} = WT \times [(\delta \times t^k_{(i,j)}) + ((1-\delta) \times kt_{(i,j)})].$$

其中:$i \in G, j \in S, 0 \le \delta \le 1, k=1,2,\ldots,\infty$;$G$ 为所有点集合,S 为所有超级点集合,$S \subseteq G$;δ是一个常数加权因子,表示等待周期对不同网络场景的敏感程度.如果 ROAD 应用在固定网络环境,如 LAN 或 Internet,$kt_{(i,j)}$变化可能由于暂时的网络拥塞引起,这种变化不应该影响 $t^k_{(i,j)}$,因此,δ可以接近 1;如果 ROAD 应用在移动网络,$kt_{(i,j)}$可能因为环境、信号质量发生变化,δ可以接近 0,表示 $t^k_{(i,j)}$ 随网络场景的状态而变化.在实际应用中,考虑到协议的计算开销,可以通过放大因子 WT 适当增大 $t^k_{(i,j)}$ 的倍数.

2.2.3 ROAD 路由算法

基于加速路由表和切换机制,我们给出了 ROAD 路由算法.图 2 为超级点 s_i 路由算法的伪代码.其中:n 为超级点数量;k 为 Qfinger 表记录数;src 为前一跳的节点 ID.

```
s_i.route(id,src)
(1)  if id∈(s_{i-1},s_i) then
(2)     forward qury(id) to finger[j]∈(s_i,id);
(3)     Return Route Succeeded;
(4)  end if
//search in two neighbor super peers through indexes
(5)  if src∈(s_{i-1},s_i) then
(6)     for j=n downto j=2 do
(7)        if id∈(s_{j-1},s_j) then
(8)           forward qury(id) to s_j;
(9)           set time = t^k_{(s_i,s_j)};
(10)       end if
(11)    end for
(12)  end if
(13)  if received return from s_j within time then
(14)  Return Route Succeeded;
(15)  end if
//search in whole super peers through super peer table
(16)  if id∉(s_{i-1},s_i) or time out then
//search in all peers using Chord protocol
(17)    for j=k downto j=1 do
(18)       if finger[j]∈(s_i,id) then
(19)          forward qury(id) to finger[j];
(20)       end if
(21)    end for
(22)  end if
```

Fig.2 Pseudo codes of lookup algorithm in ROAD

图 2 ROAD 路由算法伪代码

此算法与 Chord 算法比较可以看出:步骤(6)~步骤(15)增加了主动路由功能,并根据网络状态自适应地调整等待周期,在获得较快路由的同时没有增加任何多余的网络负载.算法的复杂度与 Chord 相同.

2.3 路由恢复

节点加入或失败的事件会导致路由表失效,从而降低路由的效率,因此,需要恢复算法定期更新路由表.

2.3.1 普通节点恢复

ROAD 恢复算法类似 Chord,即周期性地运行 $n.find_successor(n+2^{i-1})$访问 finger 表每一条记录内的节点.finger 表中有 $\log_2 N$ 条记录,每条记录的更新跳数为 $O(\log N)$,Chord 恢复算法复杂度为

$$\log_2 N \times O(\log N) = O(\log^2 N)^{[6]}.$$

为了减少恢复算法带来的开销,ROAD 在更新 finger 表时不是访问所有 finger 节点,通过访问一次所在弧的超级点表,即可更新 finger 表内所有超级点的记录.如果合适地选择超级点的数量,则可以获得比按需路由要低的维护代价.

命题 **1**. ROAD 中加速路由表的维护代价低于按需路由.

证明:令 ROAD 中存在 M 个超级点,节点总数为 N.若令 $M=\sqrt{N}$,超级点的存储容量为最低[3,4].假设 M 个超级点在网络内平均分布,则每 N/M 个点中存在一个超级点.根据定义 1,当 $g[i]$小于 N/M 个节点 ID 空间时,这条记录的节点才为普通点,即$(2^i-2^{i-1})<N/M$,每个 finger 共有 $\log_2(N/M)$条记录需要更新,这样,ROAD 中 finger 表恢复算法复杂度为 $\log_2(\sqrt{N}) \times O(\log N) = \frac{1}{2}O(\log^2 N)$,要低于 Chord 的维护代价.

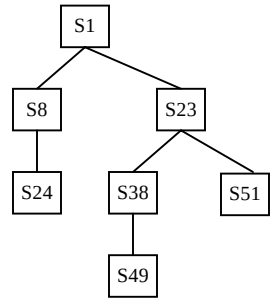
2.3.2 超级点恢复算法

主动路由对超级点依赖性大的主要原因是需要维护全局一致的超级点表.当出现节点加入或失败时,必须更新所有超级点,维护代价远高于按需路由.基于按需路由特性,我们通过改进研究[11]的 Chord 广播系统,提出了一种幂次序组播算法恢复超级点表(如图 3 所示).

```
s.multicast(update,upbound)
(1)   for i=1 to m do
(2)    if list(2^i)∈(s,upbound) then //set multicast range as (s,upbound)
(3)      new=list(2^i); // set this node to children if it is within multicast range
(4)     if list(2^(i+1))∈(s,upbound) then
(5)      new_upbound=list(2^(i+1)); //set this node as the last children
(6)     else new_upbound=upbound; //partition the new multicast range
(7)     end if
(8)      forward (update,new_upbound) to new;
(9)    else exit; //partition the new multicast range
(10)   end if
(11)  end for
```

(a) Pseudo codes of power sorting multicast

(a) 幂次序组播算法



(b) Multicast tree of super peers from Fig.1 by PSM

(b) 图 1 所示超级点按照(a)算法生成的组播树

Fig.3

图 3

含有 M 个节点的超级点表构成一个单向循环链表 $list$.对$\exists s\in list$,令 $list(1)=s$,则 s 的第 i 个邻居为 $list(2^i)$, $i=1,2,\ldots,m$.其中,m 为小于 $\log M$ 的最大整数.$\exists s\in list$ 的组播空间为$(list(2^i),list(2^{i+1}))$,其中,初始化时上限 $upbound=list(M)$.

与 Gnutella 采用的类似洪泛算法和主动路由[6,7]以及网格[11]采用的 Gossip 算法相比,幂次序组播算法不会产生重复消息,所有节点获得消息的概率高,并且每级消息扩散数量 k 和扩散轮次 i 为最低.研究[11]证明:Gossip 算法在 $k=\log M+c$ 时,消息覆盖所有节点的概率较高;并且当 $i>6$ 时,转发消息失败的次数会明显增加,即 10^7 的网络规模下无法正常工作.我们的算法 $k=\log M$, $i=\dfrac{\log M}{\log\log M}$,并且延时最低.

命题 **2**. ROAD 中幂次序组播算法消息扩散轮次和相对延时为 $O\left(\dfrac{\log M}{\log\log M}\right)$.

证明:由算法可知,$k=\log M$,即每轮消息扩散到 $\log M$ 个点.令扩算轮次为 i,则有

$$1+\log M+\log^2 M+\ldots+\log^i M=\frac{1-\log^i M}{1-\log M}=M .$$

所以,$i=\log_{\log M}M+\log_{\log M}(\log M-1)$.可以得知 $i=O(\log_{\log M}M)=O\left(\dfrac{\log M}{\log\log M}\right)$,低于 Gossip 算法的 $O(\log\sqrt{M})^2$.假设每一轮扩散最大延时 $t=\max\{t_k|k=1,2,\ldots,\log M\}$,则总延时为 $T=O\left(\dfrac{\log M}{\log\log M}\right)t$.由于幂次序组播算法在超级点之间直接转发消息,所以 $RDP=1$,因此相对延时为 $O\left(\dfrac{\log M}{\log\log M}\right)$. □

3 部分算法模拟结果

当前模拟主要集中在混合路由算法.本文基于 JDK1.4.0 实现了一个模拟器模拟 4 000 个节点平均分布在长度为 2^{12} 的 ID 空间,其中设定 64 个超级点,每个超级点管理长度为 64 的 ID 空间,finger 表的长度为 12.在两种网络波动状态下,分别比较基于超级点的单跳路由算法(One-hop),Chord 和 ROAD.

3.1 降低延时的模拟结果

波动节点分布在 4 000 个节点样本空间,确定一个波动率 r(表示每秒并发加入或退出网络的节点数),分析

平均查找率和平均查找延时,其中,查找延时用路由的跳数表示.所有路由都不启动恢复算法.图 4 显示了 r 不断增大时两种路由算法的效果.在网络波动较小时,ROAD 基本保持了常数级的延时,说明发现服务主要采用主动路由;随着 r 的增大,在发现服务失败时切换到了按需路由,从而保证了查找率.由于采用了加速路由表,有效抑制了延时的增加.我们认为,ROAD 此时处于稳定状态.

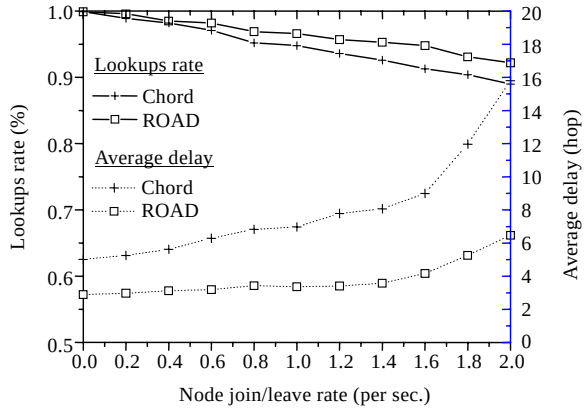


Fig.4 Performance comparison between Chord and ROAD

图 4 Chord 与 ROAD 的性能比较

3.2 改善超级点依赖性的模拟结果

为了模拟 ROAD 超级点的特性,我们设置波动节点全部分布在 64 个超级点中.图 5 显示:随着 r 的增大,One-hop 的查找率线性下降,说明路由强烈依赖超级点.ROAD 通过切换路由策略,保证了查找率的稳定;但延时增加明显.图 5 说明,ROAD 以延时为代价改善路由策略对超级点的依赖性.另外,通过幂次序组播算法可以加快超级点状态的更新,使 ROAD 恢复到稳定状态.

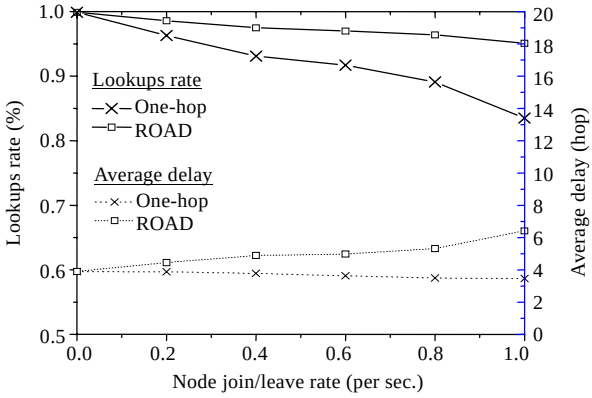


Fig.5 Performance comparison between One-hop and ROAD

图 5 One-Hop 与 ROAD 的性能比较

4 结论与未来工作

基于混合策略的思想,本文提出了一种动态环境中的 P2P 发现服务算法.它可以根据波动程度在不同的路由策略之间切换来适应不同的网络场景.由于缺乏实际的环境,我们无法测试基于 QoS 参数的切换机制的效率.下一步的工作以 ROAD 为基础,研究基于最小延时的超级点选择算法,使最小延时的超级点尽可能多地出现在路由路径上,从而提供最短延时的发现服务.我们相信:ROAD 采用不同服务质量的超级点选择算法,可以扩展成

面向不同服务的发现机制.

References:

[1] Yan F, Zhan SY. A peer-to-peer approach with semantic locality to service discovery. In: Proc. of the 3rd Int'l Workshop on Grid and Cooperative Computing. LNCS 3251, Berlin: Springer-Verlag, 2004. 831−834.

[2] Lu XC, Li DS, Wang YH, Lu K. Research on peer-to-peer distributed storage systems. Journal of Computer Research and Development, 2003,40(Suppl.):1−6 (in Chinese with English abstract).

[3] Saroiu S, Gummadi PK, Gribble SD. A measurement study of peer-to-peer files sharing systems. In: Proc. of the Multimedia Computing and Networking. 2002. http://www.cs.washington.edu/homes/gribble/

[4] Ying C, Shi ML. QoS routing in ad-hoc network. Chinese Journal of Computers, 2001,24(10):1026−1033 (in Chinese with English abstract).

[5] Stoica I, Morris R, Liben-Nowell D, Karger DR, kaashoek MF, Dabek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. IEEE Trans. on Networking, 2003. http://pdos.csail.mit.edu/chord/

[6] Gupta I, Birman K, Linga P, Demers A, van Renesse R. Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead. In: Proc. of the 2nd Int'l Workshop on P2P Systems. 2003. http://iptps03.cs.berkeley.edu/

[7] Mizrak AT, Cheng YC, Kumar V, Savage S. Structured superpeers: Leveraging heterogeneity to provide constant-time lookup. In: Proc. of the 2nd Int'l Workshop on Peer-to-Peer Systems. 2003. http://iptps03.cs.berkeley.edu/

[8] Gupta A, Liskov B, Rodrigues R. Efficient routing for peer-to-peer overlays. In: Proc. of the 1st Symp. on Networked Systems Design and Implementation (NSDI 2004). 2004. http://www.usenix.org/events/nsdi04/

[9] Gummadi K, Gummadi R, Gribble S, Ratnasamy S, Shenker S, Stoica I. The impact of DHT routing geometry on resilience and proximity. In: Proc. of the SIGCOMM 2003. 2003. http://www.acm.org/sigs/sigcomm/sigcomm2003/

[10] Garces-Erice L, Biersack EW, Felber PA, Ross KW, Urvoy-keller GU. Hierarchical peer-to-peer systems. In: Proc. of the INFOCOM 2003. 2003. http://www.eurecom.fr/~btroup/BPublished/

[11] El-Ansary S, Alima LO, Brand P, Haridi S. Efficient broadcast in structured P2P networks. In: Proc. of the 2nd Int'l Workshop on Peer-to-Peer Systems. 2003. http://iptps03.cs.berkeley.edu/

附中文参考文献:

[2] 卢锡城,李东升,王意浩,卢凯.基于对等模式的分布式存储技术.计算机研究与发展,2003,40(增刊):1−6.

[4] 英春,史美林.自组网环境下基于 QoS 的路由协议.计算机学报,2001,24(10):1026−1033.

杨峰(1972－),男,山东济南人,博士,主要研究领域为 P2P 计算,未来互联网络体系结构,分布式计算.

战守义(1940－),男,教授,博士生导师,CCF 高级会员,主要研究领域为虚拟现实,分布式计算,计算机网络.

李凤霞(1953－),教授,主要研究领域为虚拟现实仿真技术.

郑纬民(1946－),男,教授,博士生导师,CCF 高级会员,主要研究领域为 P2P 计算,分布式计算,并行计算.

余宏亮(1976－),男,讲师,主要研究领域为 P2P 计算,分布式计算,并行计算.