P2ST:基于带权搜索树的P2P搜索模型*)

吴 艾[1] 刘心松[1] 郝 尧[1] 袁连海[2]
(电子科技大学计算机科学与工程学院 成都 610054)[1]
(成都理工大学工程技术学院计算机系 乐山 614007)[2]

摘 要 针对非结构化P2P系统搜索效率低的问题,提出了一种基于K叉带权搜索树的P2P搜索模型P2ST。模型构建了服务于搜索的k叉带权树,节点按查询命中率大小在树中由上至下排列,命中率大且稳定的节点处于树的上层,搜索时可由此确定消息扩散的方向。采用缓存上层节点、建立搜索结果和发起节点索引、过热资源复制、为叶节点添加远程邻居等方法进一步提高搜索效率和平衡负载。分析和仿真结果表明,提出的模型能大量减少无效消息,具有较高的搜索效率,且维护搜索树的开销较小。

关键词 非结构化P2P,搜索模型,带权搜索树,查询命中率,索引

P2ST:A Weighted Search Tree-based P2P Searching Model

WU Ai[1] LIU Xin-Song[1] HAO Yao[1] YUAN Lian-Hai[2]
(School of Computer Science, University of Electronic Science & Technology of China, Chengdu 610054)[1]
(The Engineering & Technical College of Chengdu University of Technology, Leshan 614007)[2]

Abstract Improving search performance is an important issue in Peer-to-Peer (P2P) network systems. Although many policies are brought forward to address the issue, the question still exists. A searching model based on weighted search tree is proposed to improve search performance of unstructured P2P networks. A logical weighted k-tree is set up according to historical hit ratio, peers that have high probability to hit the query rise to higher layer of the tree, so that peers that unstable and have few hot resources are usually on low layer of the tree. Methods are also employed to increase efficiency of search, such as Hot peers caching, remote peer connections, source peer index and result index, and so on. Performance studies based on analyses and simulations are carried out, the results show that the proposed model can avoid a large amount of unnecessary messages, the overhead to maintain the tree is minimal, and outperforms related works in terms of search efficiency and search latency.

Keywords Unstructured P2P, Searching model, Weighted search tree, Hit ratio, Index

1 引言

非结构化P2P系统以其结构简单、组织方便,在大规模网络环境的资源共享中得到了最广泛的应用。资源搜索效率决定着这类系统的可用性,是系统的关键所在。简单有效的盲洪泛是非结构化P2P最初的和基本的搜索方式[1],但网络开销太大,导致系统扩展性差。随机walker搜索是非结化P2P的另一种基本的搜索方式,但无任何引导路由线索的单walker方式因为其效率低、时延太大而不能实用。

分层的方式把平面的P2P结构分为多层,上层节点管理一部分下层节点,同层节点之间互相连接。如KaZaA采用超级节[2],可在一定程度上缓解大流量问题,然而超级节点可能会造成区域内单点失效和成为系统性能瓶颈。

对资源和节点分类、分组的目的是缩小搜索范围,加快搜索速度,分类分组可根据地域、单位、兴趣或内容等进行。Xinliu等把所有P2P节点分为多个组,同时每个节点又属于不同的类,每个类形成相对独立的逻辑网络,搜索消息只在与搜索内容相关的逻辑网络中扩散,因此减小了搜索范围,能在一定程度上减少冗余流量[3]。分层常和分类方法结合使用,如MLTG[4]把节点分为super peer、deletegate和local节点,按主题对网络分层分类,不同类的节点分布于不同的层。

有选择地路由转发和引导路由能使搜索具有方向性、目的性,选择的依据可以是搜索结果索引、节点间存储内容的相似度、其他历史信息等。部分路由索引方法中[5],路由索引只维护最热门的请求内容索引,其他内容仍然采用传统的扩散方法。文[6]中采用了概率路由表来维护邻居信息,根据路由表,搜索请求能以较高的概率和较短的路径达到目的节点。WenTsuen Chen等采用基于兴趣对节点分组和引导请求的转发以减少带宽消耗[7]。

缓存和多复本策略可提高系统的可扩展性。结合均匀和按比例的资源复制方法,E. Cohen等提出了三种复本策略[8],通过控制副本的放置位置而减小搜索范围,根据请求比率决定复本的分发比率。

上述P2P系统提出了多种提高搜索效率的方法,但仍存在重复消息多、搜索效率低或时延大等问题。对于此,提出基于K叉带权搜索树的非结构化P2P搜索模型P2ST。模型构建了基于K叉树的搜索结构,采用升降级机制,每个节点在树中的位置按查询命中率为权值进行排列,权值越大的节点

*)本课题得到四川省应用基础研究项目(编号04JY029-017-2)基金资助。吴 艾 博士生,主要研究方向为分布式计算;刘心松 教授、博士生导师,研究方向为宽带网络、分布并行处理、操作系统和数据库等。

处于树的越上层，使系统中的节点有规律地分布，越上层的节点拥有热点资源的概率越大。并采用建立搜索结果历史索引和搜索发起节点索引、缓存上层节点、部分过热资源复制和为叶节点增加远程邻居等方法，进一步提高搜索效率和平衡负载。分析和仿真结果表明：P2ST 能大量减少无效流量、提高搜索效率，且搜索树的维护开销很小。

2 系统模型

通常的非结构化 P2P 系统拓扑结构为具有幂律(power-law)度分布和高聚集度特性的逻辑网络，称为原网络。在维护系统结构的完整、可靠和容错方面，原网络具有良好的性能，因此仍然保留原网络的存在。在树结构中，节点的邻居只有父、子节点和很少量的冗余节点，节点的度较小，系统结构维护开销也很小，在扩散搜索请求时不会产生重复消息。而基于原网络的搜索效率低、无效流量大，因此在原网络之外构建以 K 叉树为基础的逻辑结构，用于资源搜索。进一步优化树的搜索特性，给树中每个节点赋予权值，且对节点按权值有规律地排列，则得到 K 叉带权搜索树(KWST, K Weighted Search Tree)。

定义 1 称一棵树为 K 叉带权搜索树，如果满足条件：(1)每个节点最多有 K 个子节点；(2)只有树的最底层和次底层存在叶节点和子节点数小于 K 的节点；(3)对任意节点，其权值小于等于父节点的权值，同时大于等于所有子节点的权值。

以 KWST 为基础的 P2P 搜索模型，称为 P2ST。设 P2ST 系统有 N 个节点，序号为 i 的节点表示为 C_i，C_i 的子节点集、兄弟节点集和父节点分别表示为 son(i)、brother(i)和 father(i)。

为保证 P2ST 搜索树结构的可靠性和容错能力，除父子节点外，每个节点还有少量连接备份节点。定义根节点和第一层节点两两互为备份节点，其他节点的备份节点为祖父节点。初始化和重构过程中的搜索树可能不满足定义 1 的条件(3)，但仍称为 KWST。

P2ST 的搜索树可以和原网络同时构建，也可以在已经存在的 P2P 网络上单独建立，树的初始化算法如下。

算法：初始化构造搜索树

相关术语：newPeer：新到达节点；tmpPeer：树中已存在的任一节点；sonNum(C_i)：计算 C_i 的子节点数；subTreeFull(C_i)：以 C_i 为根的子树是否为满树

```
ConstructKWSTree(newPeer, tmpPeer){
    if newPeer is the first Peer then
        root = newPeer
        return success
    end if
    if subTreeFull(tmpPeer) == True || sonNum(tmpPeer) = 0
then //is leaf or is full
     tmpPeer = fahter(tmpPeer)
     while subTreeFull(tmpPeer)== True
          tmpPeer = fahter(tmpPeer)//子树已满，向上
     }
    end if
    while sonNum(tmpPeer)== k{//向下寻找未满子树
          if sonPeer∈son(tmpPeer)&&
               subTreeFull(sonPeer)! = True && sonNum(son
               Peer)>0 then //选一未满子节点
               tmpPeer = sonPeer
          end if
    }
    newPeer add in tmpPeer
    return success
}
```

说明：如果出现整棵树全满的情况，则新节点可以加入到任意叶节点。

如节点退出，可能在树中形成空缺而影响树的完整性，采用父节点代理的方法，快速维护树的整体结构。方法是如 C_i 发现子节点 C_j 退出，且 C_j 非叶节点，则 C_i 暂时代理 C_j，由于 C_j 的子节点以 C_i 为备份节点，所以系统很快恢复而不会影响树的连通性，之后 C_i 在 C_j 的子节点中选择一个接替 C_i 的代理位置。

3 搜索模型

3.1 思想

以历史查询命中率为 P2ST 中节点的权值，当系统的结构趋于稳定，从树的上层往下，节点的命中率和稳定性逐渐减小，上层节点的热点资源较多，而下层的非热点资源较多，因此 P2ST 提供了较为确定的资源冷热分布情况。任意节点都可以很容易地确定热点区域的方向，而且可在很短时间内到达。对于拥有非热点资源的节点，采用远程连接加强它们之间的联系，同样可以提高冷门资源搜索效率。同时采用部分热点资源复制、索引和上层节点缓存机制，进一步提高了搜索效率的同时，解决因树型结构而导致的部分节点负载较重的情况。新节点都从低层加入，频繁出入系统、不稳定的节点总是处于树的较低层，因此系统搜索性能的稳定性也能得到保证。

3.2 查询命中率

P2ST 中，节点的权值为查询命中率(HR, Hit Ratio)，HR 为从节点加入系统开始到当前时刻的查询命中率综合评价值，反映节点拥有的资源在网络中的热度。

设节点到达系统时刻为 0，每个时间段长度为 τ。在第 j 时间段内，到达 C_i 的请求次数为 QueryNum，其中命中的次数为 HitNum，以 $R_i(j)$ 表示在第 j 时间段内 C_i 的命中率。

$$R_i(j) = HitNum / QueryNum \tag{1}$$

$W_i(j)$ 为 C_i 在 $[0, j\tau]$ 时间内的查询命中率，递归定义如下：

$$W_i(j) = \alpha W_i(j-1) + (1-\alpha)R_i(j),\ 0 \leqslant \alpha \leqslant 1,\ j > 0 \tag{2}$$

当 j 等于 0 时，初始值 $W_i(0)$ 定义为

$$W_i(0) = 0 \tag{3}$$

α 为命中率系数，取不同的值，反映查询命中率更侧重于长期或者是近期结果，α 越小，最近时间段的命中率越重要。C_i 最新的查询命中率简单表示为 W_i。

命题 1 对任意节点 C_i，$W_i(j)$ 大于等于 0，小于等于 1

证明：由递归方程求解可得 $W_i(j)$ 的解析式为

$$W_i(j) = (1-\alpha)\sum_{k=0}^{j-1}\alpha^{j-1-k}R_i(k)$$

因为 $0 \leqslant \alpha \leqslant 1$，$0 \leqslant R_i(k) \leqslant 1$，所以当 $R_i(s) = 0$，$0 \leqslant s \leqslant j-1$ 时，$W_i(j)$ 取得最小值 0，当 $R_i(s) = 1$，$0 \leqslant s \leqslant j-1$ 时 $W_i(j)$ 取得最大值

$$W_i(j) = (1-\alpha)\sum_{k=0}^{j-1}\alpha^{j-1-k} = 1-\alpha^j,\ \text{可得 } W_i(j) \leqslant 1$$

所以 $0 \leqslant W_i(j) \leqslant 1$，得证。

因为请求在树型结构中节点的邻居之间转发，相邻节点收到的请求次数相近，所以邻近节点的 HR 具有可比较性，HR 能体现节点资源对请求的满足程度。

3.3 搜索树的升降级算法

为满足定义 1 中条件(3)，当节点的 HR 值发生变化或系统初始化过程中，需要对节点在树中的位置做相应的调整，调整过程称为升降级过程，对应 RS(Rise and Sink)算法。节点从当前层移动到上一层，称为升级，反之为降级。

对 C_i，其父节点为 C_f，兄弟节点 $C_b \in$ brother(i)，子节点

$C_s \in son(i)$，每个节点维护一张包括子节点和自身的 HR 列表(HRL，HR List)。RS 算法如下。

算法：搜索树的升降级

Step1：C_s 如发现计时器 Timer 变为零，计算最新的查询命中率 W_s，并通过消息 NEW_RATIO 发送给 C_i，同时更新 HRL，重设 Timer。

Step2：C_i 收到 C_s 的 NEW_RATIO 消息，取出消息中的 W_s，更新 HRL。查看 HRL 中所有子节点的 HR，如果 W_i 仍最大，则完成；否则进行升降级处理，假设 C_s 的 HR 最大，发送 REQ_UPGRADE 消息给 C_s，请求升降级处理。

Step3：C_s 收到消息 REQ_UPGRADE，如同意升级，向 C_i 返回 AGR_UPGRADE；否则返回 REJ_UPGRADE。

Step4：C_i 若收到 REJ_UPGRADE，则完成；否则开始升降级操作：C_s 以 C_f 为新父节点，并以 C_i 和{son(i)－C_s}中节点为子节点，原 C_s 的子节点以 C_i 为新父节点。升降级完成。

算法说明：一个节点不能在连续的 2 个时间段内进行升降级操作，是为了防止抖动和树结构剧烈地变化。

升降级是逐渐优化搜索树的过程，当 C_i 的 HR 高于兄弟和父节点的 HR 时，C_i 与父节点交换位置，升级一次。新加入搜索树的节点总是作为叶子节点而位于树的底层或次底层。当系统运行一段时间后，搜索树的结构趋于相对稳定的状态，稳定可靠且 HR 较高的节点总是能位于树的上层，稳定但 HR 低的节点和不稳定但 HR 高的节点将位于树的相对中间层，而不稳定且 HR 低的节点位于树的较低层。

3.4 搜索算法

名词术语：sourcePeer，搜索发起节点；query：搜索请求；curPeer，当前节点；lastPeer：相对于 curPeer 的搜索路径中的前一个节点；nextPeer：相对于 curPeer 的搜索路径中的下一个节点 destPeer：搜索成功的节点。

3.4.1 搜索树中的基本扩散

在系统运行的初始状态或节点加入系统的短时间内，没有可利用的历史搜索经验数据，或对于其他需要的情况(如非热点资源)，采用基于扩散的搜索，将请求转发给所有邻居，当前节点的容错备份的节点不参与搜索。搜索停止的条件为 TTL 为 0 或搜索成功或已没有可转发的节点。

采用基本的搜索方法不会产生重复消息，且搜索时延(跳)的上限为两倍树高。随着系统运行时间的增加，P2ST 搜索树逐渐变得有序和相对稳定，节点积累了一定的搜索经验信息，因此可对搜索算法优化。

3.4.2 搜索结果索引缓存

本文采用一种基于搜索结果索引的缓存策略来提高搜索效率。每个节点维护搜索结果索引表(RIL，Result Index List)，其内容为搜索结果摘要：请求资源和对应的成功返回结果的节点。新索引项的建立算法：

Step1：destPeer 搜索成功后，首先将完整结果返回给 sourcePeer，然后将搜索结果摘要按搜索路径原路逐跳返回。

Step2：如路由节点 CurPeer 收到结果摘要，查看 RIL，如 RIL 中没有，则建立新索引项；如该摘要已存在，则按规则更新 RIL。将摘要继续返回给 lastPeer。

Step3：路由节点如在超时时限内没有收到结果摘要，则认为转发的 query 在 nextPeer 搜索失败，如果 RIL 缓存了对应于 query 和 nextPeer 的索引项，则认为该索引项失效，删除之。

说明：如 RIL 已满，按 LRU 算法更新。

3.4.3 搜索源节点索引

如节点 C_i 发起请求搜索资源 Q，如成功得到结果，C_i 往往会存储或缓存资源 Q，因此如 C_j 曾转发过 C_i 的 Q 搜索请求，则 C_j 可认为 C_i 有较大概率拥有 Q 资源。

因此采用缓存发起节点提高搜索成功概率。每个节点维护一张源节点表(SPL，Source Peer List)，内容为历史上到达的搜索发起节点和请求的资源列表，用 LRU 策略更新。

3.4.4 搜索算法描述

在基本扩散和缓存索引的基础上建立搜索算法，在某节点的搜索顺序为结果索引列表 RIL、源节点索引列表 SPL 和本地存储。如果不成功则扩散到邻居，描述如下。

```
Function SearchProcess( query ){
    destPeer = RetriveRIList (query, RIList)// RIList 中查询
    if destPeer == Null Then
        destPeer=RetriveSPList(query,SPList)//查询 SPList
    endif
    UpdateSPList (query, destPeer)//更新 SPList
    if destPeer ! = Null
        SendTo(query, destPeer)//转发给 destPeer
        UpdateRIList(query, destPeer)//更新 RIList
        ModifyRecentHitRatio()//修改最近命中率
        return TRUE
    endif
    if RetrieveLocal(query)! =NULL then //本节点搜到
        SendTo(resultSet, sourcePeer)//返回给发起节点
        SendTo(resultSetAbstract, lastPeer)//原路返回结果
        ModifyRecentHitRatio()
        return TRUE
    endif
    if TTL > 0 then //转发请求
        SendTo(query, all neighbors except lastPeer)
        return FALSE;
    endif
}
```

下面采用几种方法来优化存在的问题：下层节点离热点区域较远，部分热点节点访问量过大。

3.4.5 缓存上层节点

由于上层节点存储有较多的热点资源，因而成为热点区域，在此区域搜索成功的概率较大，但较下层的节点离该区域较远，因此下层节点维护一张上层节点表(HLL，High Layer List)，在转发消息前，查看 HLL 中是否有节点存在，如果有，则优先把请求扩散到这些节点，而不必逐级向上扩散，既保证了搜索效率又减小了时延。

3.4.6 添加远程节点

在树结构中，对于距离最远的节点来说，搜索消息需要多跳才能达到，使得难于搜索到某些非热点资源，或者即使搜到，付出的代价也较大。根据小世界理论，如果在规则网络中加入较少的远程边，能大大缩小节点之间的平均距离，同时考虑到 K 叉树的最远距离在叶节点之间形成，因此，为叶节点添加少量远程叶节点。远程边的添加是在搜索过程中发现并连接的。

通过远程节点搜索，既可以减小搜索时延，又增加了非热点资源的搜索成功率。在搜索非热点资源时，可采用非对称 TTL 设置，将向上层转发的消息的 TTL 设置为较小值，而向下层传递的搜索消息的 TTL 值设为正常值，从而使搜索消息更多地向下层节点转发。

3.4.7 过热资源复制

在树中搜索可能会造成部分上层的节点被访问的次数过多、负载过重的问题，对于此，采用对过热资源进行复本复制的方法。当某些资源在最近一定时间内的访问频率超过了阈值，且资源的大小小于限定值时，如搜索成功，搜索发起节点复制该资源，以减轻该资源所在节点的访问压力，且热点资源复本数的增加可进一步提高搜索效率。

4 性能评测

4.1 性能分析

设树节点数为 N，最大子节点数为 k，树高度为 h，则下式成立：

$$\frac{k^h-1}{k-1}+1\leqslant N\leqslant\frac{k^{h+1}-1}{k-1}$$

因此可得

$$\log_k[(k-1)N+1]-1\leqslant h\leqslant\log_k[(k-1)N-k+2]$$

所以搜索树高度 h 为

$$h=\lfloor\log_k[(k-1)N-k+2]\rfloor \quad (4)$$

节点间的最大距离 D 小于等于 2 倍树高，即

$$D\leqslant 2\lfloor\log_k[(k-1)N-k+2]\rfloor \quad (5)$$

D 随 N 以 $O(\log N)$ 增加。

搜索树维护开销分析：研究表明，节点数为 N 的典型的 P2P 网络平均每分钟有 $0.004N$ 个节点出入网络[9]。每次节点出入系统，需从发生节点到达叶节点进行相关处理，而计算可得任意节点到叶节点的平均距离为 $h/(k-1)$，则每分钟网络中产生 $0.0044Nkh/(k-1)$ 个与搜索树更新相关的消息，因此每分钟的更新消息量级为 $O(N)$。而 P2P 网络中平均每节点每分钟发出 0.3 个搜索请求[9]，每个请求扩散会引起 $O(N)$ 个请求消息，因此网络中每分钟由于搜索扩散产生的消息为 $0.3O(N^2)$。所以，更新搜索树的开销与搜索操作的开销相比非常小，可以忽略不计。

4.2 仿真实验

采用 GT-ITM[10] 工具生成原网络，并同时生成 P2ST 搜索树，其中 k 为 3，节点数 N 为 5000。资源数量为 2000，每个资源的复本数为 5，按 power-law 分布到到各节点，约 80% 的资源分布到约 21% 的节点。资源被访问频率服从 Zipf(子数参数为 0.95)分布，搜索发起节点随机选取，ttl 取值为 4。每节点的索引表的容量为 20 条。系统中每发生 500 次查询则记录一次搜索结果，更新一次查询命中率，命中率系数 α 取 0.5。实验时间以系统中发起的查询消息数来表示，实验总时间为所有节点共发起 50 万个查询的时间。

对比系统为原网络 P2P 模型(OriginP2P)和最近提出的多 walker 的搜索模型[11](K-WalkerP2P)，与 P2ST 具有相同的系统规模、资源分布和访问规律，均采用缓存和索引机制来提高搜索效率，K-WalkerP2P 的 Walker 数量为 4，且 K-WalkerP2P 还利用命中率权值来引导路由。

评测参数：(1)搜索效率 η，为成功搜索次数与消耗的资源的比值，消耗资源以参与搜索的节点数表示；(2)时延 Delay，指搜索开始至收到第一个返回结果的时间；(3)节点负载 Load，在较长的一段时间内，每个节点参与搜索的次数，包括转发消息和搜索。

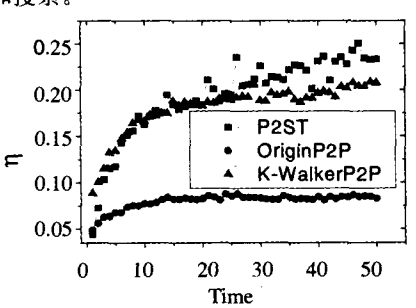

图 1 P2ST、OriginP2P、K-WalkerP2P 的搜索效率对比

图 1 为 P2ST 和对比系统的 η 随时间变化情况。在系统初始运行的一小段时间，由于 P2ST 的节点随机分布于树中，搜索树的热点还没有达到有规律的分布，而 K-WalkerP2P 采取了缓存和索引提高效率，并且优先向高命中率节点转发，所以在系统的初始化阶段略高于 P2ST。之后，P2ST 的节点逐渐按命中率高低从上至下规则地排序，搜索的方向性变得明确，树的状态也趋于稳定，所以 P2ST 的 η 很快升高并超过 K-WalkerP2P，且运行时间越长优势越明显。OriginP2P 虽然同样采取了缓存和索引，但搜索的方向性仍然较差，其扩散的方式产生较多的无效消息，所以其效率最低。

图 2 为发起节点收到第一个返回结果的时间随系统时间变化情况。P2ST 具有最小时延，这是因为 P2ST 的搜索方向性最明确，大部分消息能在很快到达热点区域，从而提高了找到结果速度和几率。而 K-WalkerP2P 由于采用有选择性地转发消息，其时延也小于 OriginP2P。

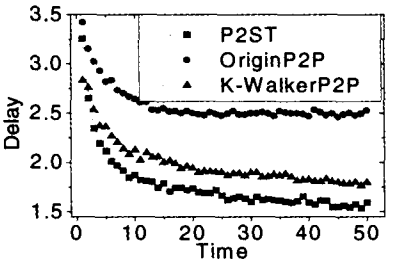

图 2 P2ST、OriginP2P、K-WalkerP2P 的搜索成功的时延对比

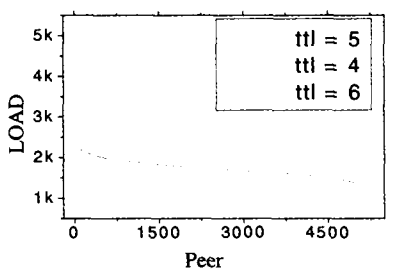

图 3 P2ST 的负载分布

图 3 为 P2ST 的节点负载分布情况，节点按负载从高到低排列，x 轴为节点。可看到，对相同的 TTL，节点负载变化平缓，有一小部分节点负载稍高，但并没有出现少量节点负载极高的情况，最高节点负载为平均节点负载的 2 倍，是在可以接受的范围。这是由于采用了缓存索引搜索结果和缓存部分热点节点的原因，使各节点的负载趋于相对均衡。

同时仿真了节点退出、加入系统对搜索效率的影响。当系统运行一段时间后，按接近实际系统的情况，系统中每发生 60 次查询，有一个节点出入系统，且节点退出时不通告任何其他节点，考虑了 2 种情形：(1)Random Peer Exit，退出节点随机选择；(2)Hot Peer Exit 退出节点从访问率较高的热点节点(占总节点数 20%)中选择一个。与无节点(No Exit)出入系统情况相比较。如图 4，在有节点退出之初，搜索效率下降明显，但在较短时间内，停止下降趋势，并逐渐回升，2 种情况的回升速度相差不大。这说明 P2ST 的搜索效率并不因为热点节点的退出而受到较大的影响，且节点退出的影响很快得到恢复。

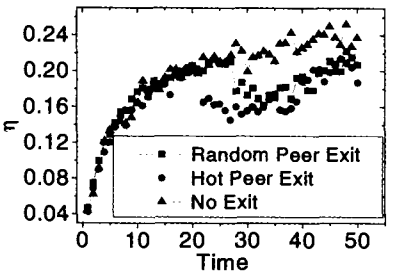

图 4　节点退出、加入系统对 P2ST 的搜索效率的影响

结论　为提高非结构化 P2P 系统搜索效率，提出了一种基于 K 叉带权搜索树的 P2P 搜索模型 P2ST。P2ST 构建了服务于搜索的 K 叉带权树，定义和计算综合的查询命中率评估值，并按评估值动态调整节点在树中的位置，命中率大的节点向树的上层移动，且新节点从叶节点加入，因此命中率大且状态稳定的节点处于树的较上层，节点在树中有规律地分布使搜索具有明确的方向性。采用缓存上层节点、建立搜索结果摘要索引和发起节点索引、复制部分过热资源、为叶节点增加远程邻居等方法进一步提高搜索效率。分析和仿真结果表明：提出的搜索模型能大量减少无效流量；与同样采用缓存和索引等策略的 OriginP2P 和 K-WalkerP2P 相比具有更高的搜索效率和更小的时延，且维护搜索树的开销很小。

参 考 文 献

1　Matei R, Iamnitchi A, et al. Mapping the Gnutella Network. IEEE Internet Computing, 2002, 6: 50～57

2　Leibowitz N, Ripeanu M, et al. Deconstructing the Kazaa network. In: WIAPP 2003, 2003. 112～120

3　Xin liu, et al. A category overlay infrastructure for peer-to-peer content search. In. IEEE IPDPS'05, 2005

4　Chen SyuYang, T Wen-Hsien, et al. A Multilayer Topic-Group based P2P Network. IEEE AINA, 2006

5　Stephane B, et al. Exploiting local popularity to prune routing indices in peer-to-peer systems. DEXA'05, 2005

6　Kumar A, Xu J. Efficient and scalable query routing for unstructured peer-to-peer networks. INFOCOM 2005

7　Chen Wen-Tsuen, Chao Chi-Hong, et el. An Interested-based Architecture for Peer-to-Peer Network Systems. IEEE AINA, 2006

8　Cohen E, Shenker S. Replication strategies in unstructured peer-to-peer networks. In Proceedings of ACM SIGCOMM'02, 2002

9　Liu Yunhao, Liu Xiaomei, et al. Location-aware topology matching in P2P systems. INFOCOM 2004, 2004. 2220～2230

10　Calvert K, Zegura E. GT-ITM: Georgia Tech Internetwork topology models. http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html

11　冯国富，毛莺池，等. PeerRank：一种无结构 P2P 资源发现策略. 软件学报，2006(5)

（上接第 54 页）

2）并行路径的概率分布平均度：当并行支路被使用的可能性趋向等概率时，$|D_{s-t}|$ 增大；

3）路径分段数目：当并行路径数目保持不变时，增加不相关的分段数目，会令 $|D_{s-t}|$ 增大。

结束语　本文讨论了一种端到端网络路径多样性的测度，目的在于为各种不同的拓扑结构提供定量测量和比较方法。该测度基于链路被使用的概率分布，采用了类似信息熵的计算方式，其特点总结如下：

1）适用于可以抽象为有向无圈图的一类端到端网络，在给定拓扑结构和链路使用的概率分布的情况下，测度是唯一的；

2）计算之前需要预先知道网络的拓扑结构，并将该拓扑结构转换成相应的 QHD 表示；

3）将沿源点到汇点的分段路径视为随机过程，根据各段链路被使用概率的分布计算每段的多样性指数，并用所有分段随机变量的联合熵 $|D_{s-t}|$ 作为衡量整个网络多样性的大小。

路径多样性是体现网络容错能力、抗攻击能力的的重要指标，也是抵御未知类型攻击的主要手段之一，本文提出的路径多样性测度可以用来对各种不同的端到端网络的路径多样性进行比较，以利于设计出鲁棒性高的网络拓扑结构，提高抵抗攻击的能力。需要注意的是，由于一般的端到端网络包含多个层次，各个层次的拓扑结构会有差别，因此，对各个层次的路径多样性需要分别进行计算。

参 考 文 献

1　Andersen D, Balakrishnan H, Kaashoek F, Morris R. Resilient overlay networks: ACM Press New York, NY, USA, 2001

2　Han J H, Jahanian F. Impact of path diversity on multi-homed and overlay networks. In: 2004 International Conference on Dependable Systems and Networks, Proceedings. Los Alamitos: IEEE Computer Soc, 2004. 29～38

3　Tang C, McKinley P K. Improving multipath reliability in topology-aware overlay networks. In: Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on, 2005. 82～88

4　Habib A, Chuang J. MMS: A multihome-aware media streaming system. In: S. Chandra and C. Griwodz, eds. Multimedia Computing and Networking 2006, vol. 6071, Proceedings of the Society of Photo-Optical Instrumentation Engineers (Spie), Bellingham: Spie-Int Society Optical Engineering, 2006. 7106～7106

5　Teixeira R, Marzullo K, Savage S, Voelker G M. Characterizing and measuring path diversity of internet topologies. In: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2003. 304～305

6　Smith P. BGP multihoming techniques. NANOG 23, 2001

7　Akella A, Maggs B, Seshan S, Shaikh A, Sitaraman R. A measurement-based analysis of multihoming. In: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, 2003. 353～364

8　Han J, Watson D, Jahanian F. Topology aware overlay networks. In: IEEE Infocom 2005. the Conference on Computer Communications, Vols 1-4, Proceedings, Ieee Infocom Series, K. Makki and E. Knightly, Eds. Los Alamitos: Ieee Computer Soc, 2005. 2554～2565

9　de Launois C. Leveraging internet path diversity and network performances with IPv6 multihoming. http://www.info.ucl.ac.be/people/delaunoi/diversity/. 2004

10　Teixeira R, Marzullo K, Savage S, Voelker G M. In search of path diversity in ISP networks. In: Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement, 2003. 313～318