第35卷 第14期 计 算 机 工 程 2009年7月
Vol.35 No.14 Computer Engineering July 2009
·博士论文· 文章编号：1000—3428(2009)14—0030—02 文献标识码：A 中图分类号：TP393

一种负载感知的结构化P2P协议

熊 伟[1,2]，谢冬青[2]，刘 洁[3]，周再红[1]
(1. 湖南大学计算机与通信学院，长沙 410082；2. 广州大学计算机科学与教育软件学院，广州 510006；
3. 广州大学实验中心，广州 510006)

摘 要：提出并建立一种负载感知的结构化P2P协议——LaChord，采用负载感知的被动式路由表维护机制和负载感知的路由算法，通过增加轻载节点的入度来增大轻载节点作为路由中继节点的概率，在保证查询性能的前提下使消息路由到轻载节点。实验表明，与没有采用负载感知的协议相比，采用负载感知的结构化P2P协议可使系统内节点负载达到更好的平衡，负载感知的算法有助于提高系统的扩展性能。
关键词：对等网络；负载感知；结构化覆盖网；负载平衡

Load-aware Structured Peer to Peer Protocol

XIONG Wei[1,2], XIE Dong-qing[2], LIU Jie[3], ZHOU Zai-hong[1]
(1. School of Computer and Communications, Hunan University, Changsha 410082; 2. School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006; 3. Experimental Center, Guangzhou University, Guangzhou 510006)

【Abstract】 This paper proposes and builds a load-aware structured Peer to Peer(P2P) protocol——LaChord. LaChord uses a reactive load-aware routing state maintenance strategy and a load-aware routing algorithm, the former increases the probability of the light loaded nodes as the intermediate nodes forwarding messages through improving the in-degree of the light loaded nodes, and the latter makes messages routed bias to light loaded nodes with provable query performance. Simulation results indicate load-aware protocols that implement reactive load-aware routing state maintenance algorithm and load-aware routing algorithm have a better load balance than the traditional P2P protocols, so load-aware algorithms can improve the scalability of systems.
【Key words】 Peer to Peer(P2P) network; load-aware; structured overlay network; load balance

目前，路由表大小为 $O(\log N)$++的结构化P2P协议在路由表维护机制和路由算法中均没有考虑节点的负载信息。本文提出一种负载感知的结构化P2P协议——LaChord，与传统的路由表大小为 $O(\log N)$++的结构化P2P协议的不同之处在于，LaChord采用负载感知的被动式路由表维护算法和负载感知的路由算法，考虑了节点的负载信息。

1 负载感知协议LaChord

在LaChord中，每个节点采用哈希函数SHA-1生成唯一的ID，所有ID组成一个环型空间。关键字 *key* 映射到顺时针方向最近的节点，该节点记为 *successor*(*key*)。为保证消息路由的一致性，每个节点的路由表保存1个大小为16的leaf set，即8个后继节点和8个前驱节点。在Chord[1]中，假设节点ID的长度为 m 位，则每个节点把键值空间划分为 m 个切片(Slices)，Slices以成倍的大小增大；在LaChord中，节点把键值空间划分为 $2\times(m\text{-}1)$个Slices，前 m-1个Slices与Chord中的前 m-1个Slices划分相同，后 m-1个Slices与前 m-1个Slices相互对称。LaChord的键值空间切片如图1所示，节点 x 的ID长度 m 为5，共有8个路由表入口。每个路由表入口保存的有效指针至少为2。

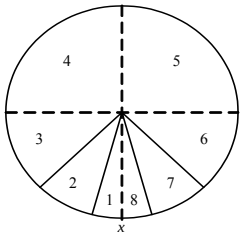

图1 LaChord的键值空间切片

1.1 负载感知的路由算法

LaChord采用单路递归路由算法，由于每个节点均把键值空间划分为对称，因此搜索方向可以采用顺时针逼近或逆时针逼近方法，节点每次搜索时首先随机选择一个方向后再进行搜索。顺时针方向逼近的路由算法伪代码如下：

L_n: the leaf set of node n
R_n: set of nodes in the routing table of node n
R_n^i: set of nodes in the ith entry of the routing table of node n
S_n^i: the key space of the ith entry of the routing table of node n
n.route(msg , key)
1: if (key ∈ [L_n.leftmost, L_n.rightmost])
2: nextHop =pick j∈L_n ∧ j is most closely follows key
3: else
4: nextHop =pick j∈R_n ∧ j is most closely precedes key
5: dest_entry = pick i∈[1,2m-2] ∧ key∈S_n^i;
6: greed_entry = pick i∈[1,2m-2] ∧ nextHop ∈S_n^i;
7: if (dest_entry== greed_entry)
8: nextSet ={ j| j∈$R_n^{dest\_entry}$ ∧ j∈(n,nextHop]};

基金项目：国家自然科学基金资助项目(60673156)；教育部科学技术研究基金资助重点项目(105129)
作者简介：熊 伟(1977－)，男，博士，主研方向：分布式计算，对等网络；谢冬青，教授、博士生导师；刘 洁，硕士；周再红，博士
收稿日期：2008-11-18 **E-mail：**freeprogman@hotmail.com

```
9:              nextHop = pick the lightest loaded node in nextSet ;
10:         endif
11:     endif
12:     if ( nextHop ≠ n ∧ nextHop ≠ null )
13:         send msg to nextHop;
14:     else
            // if nextHop==null, discard msg
15:         n is the destination of the msg;
16:     endif
```

具体说明如下：

语句 1、语句 2：如果查询的关键字 *key* 在节点 *n* 的 leaf set 的空间[L_n.*leftmost*, L_n.*rightmost*]中(其中，L_n.*leftmost* 表示节点 *n* 的 leaf set 中最远的前驱节点；L_n.*rightmost* 表示最远的后继节点)，则直接在 leaf set 空间中寻找 *successor*(*key*)。

语句 4：否则节点 *n* 首先在路由表中用贪婪式查找 *key* 的最近前驱节点 *nextHop*。

语句 5、语句 6：查找目标 *key* 和 *nextHop* 所属的节点 *n* 的路由表入口编号 *dest_entry*(), *greed_entry*()。

语句 7~语句 10：如果 *key* 和 *nextHop* 属于同一个路由表入口区间，获取可以作为下一跳节点的节点集合 *nextSet*，并在 *nextSet* 中选择负载最小的节点作为路由的下一跳节点。

语句 12、语句 13：如果下一跳节点 *nextHop* 不是本地节点 *n* 并且不为空，则把消息发送给下一跳节点。

语句 14~语句 16：否则消息已经到达目标节点 *n*，由本地节点 *n* 最终处理消息。

逆时针方向逼近的路由算法与顺时针方向逼近的路由算法相似，在此省略。由此可知，LaChord 在保证路由效率为 $O(\log N)$的前提下，使消息尽量由负载低的节点来处理。

1.2 负载感知的被动式路由表维护机制

早期研究认为，节点的路由表指针数量越少，网络维护开销越小，因此，早期研究的大多数结构化 P2P 协议是在保证查询性能为 $O(\log N)$的前提下，如何最小化节点路由表指针数量来减少网络的维护开销，这些协议的路由表大小固定、不灵活，为保证路由指针的有效，节点周期性查询适合作为路由表指针的节点或 Ping 路由表中存在的节点，是一种主动式的路由表维护策略。被动式路由表维护策略则充分把节点的路由表维护融入到查询算法中，只有当路由表中的节点数量较少时才启动主动探测算法，这种路由表维护机制适合于节点路由表大小为 $O(\log N)$++的结构化协议。在查询负载很高时，绝大多数的路由表入口均可以学习到足够的有效路由表指针，节点须主动探测学习新节点作为路由表指针的概率较小，因此，这种被动式路由表维护策略可以大大节约系统维护开销。目前已有很多采用被动式路由表维护机制的结构化 P2P 协议，例如 EpiChord[2], Accordion[3]等，但这些协议在消息路由和路由表维护中不考虑节点的负载信息，与这些采用被动式路由表维护机制的结构化协议不同的是，LaChord 使用一种负载感知的被动式路由表维护机制，这种负载感知的被动式路由表维护包括以下几种机制：

(1)在消息路由过程中，消息把途经节点记录下来，每个途经节点把消息已携带的节点加入到自身路由表中，然后把自身负载信息加入消息发送给下一跳节点。当查询消息到达目标节点，目标节点把途经节点加入消息发送给查询源节点，查询消息源节点把途经节点加入到自身路由表。每个途经节点均把消息源节点的信息加入到自身路由表。例如，假设节点 *a* 发送一个查询消息，该消息依次经过中间节点 *b*、节点 *c* 最终到达目标节点 *d*，则 *a* 可以学习到节点 *b*、节点 *c*、节点 *d*；节点 *b* 学习到节点 *a*；节点 *c* 学习到节点 *a*、节点 *b*；节点 *d* 学习到节点 *a*、节点 *b*、节点 *c*。

(2)当节点 *a* 发送消息给节点 *b* 时，节点 *a* 把本地负载信息加入消息发送给节点 *b*，节点 *b* 收到节点 *a* 的消息后，如果节点 *b* 的路由表中存在节点 *a*，则更新节点 *a* 的负载信息并盖上时间戳，如果节点 *b* 的路由表中不存在节点 *a*，则把节点 *a* 加入到自身路由表并盖上时间戳，然后，在 *ack* 消息中加入自身负载信息发送给节点 *a*，节点 *a* 更新路由表中保存的节点 *b* 的负载信息并盖上时间戳。每个路由表指针的过期时间为 2 min，即如果当前时间与路由表指针的时间戳相差大于 2 min 时，该路由表指针将被清除出路由表。

(3)节点每分钟检查路由表的每个入口，当某个路由表的入口中包含的有效路由指针数目少于 2 时，则启动主动探测算法查找该入口对应的键值空间的中间位置的节点，在该目标节点的路由表中选取至多 4 个属于该键值空间并且负载最小的节点返回给主动探测节点。

1.3 节点的加入与退出

假设加入节点 *n* 在准备加入网络之前知道一个已经存在网络中的存活节点 *n*’，由节点 *n*’发起查询 *successor*(*n*)的消息，当查询消息到达节点 *successor*(*n*)，节点 *successor*(*n*)把自身路由表和 leaf set 复制发送给加入节点 *n*，节点 *n* 通知路由表和 leaf set 中的节点已经加入网络。在通知其他节点时，加入节点 *n* 的负载采用其路由表中节点的负载的平均值。当加入节点 *n* 的路由表某个入口的有效指针数量少于 2 时，则发起查询该路由表入口指针的查询请求。

当节点 *a*、节点 *b* 在通信后，节点 *a*、节点 *b* 的路由表均保存有对方，因此，当节点 *n* 主动退出网络时，节点 *n* 通知其路由表及 leaf set 中的节点，被通知的节点将从路由表和 leaf set 中删除节点 *n*。

为维护 leaf set 的一致性，每个节点周期性探测后继节点，当节点在 30 s 内没有收到前驱节点的探测消息，则主动探测前驱节点，当发现前驱节点失败，通知其 leaf set 中的其他节点。

2 模拟实验与对比

实验比较采用负载感知算法的结构化 P2P 协议和采用传统贪婪式查找算法的结构化 P2P 协议间负载分布的不同。本文建立一种与 LaChord 相似的协议——BiChord，BiChord 协议与 LaChord 协议具有相同的拓扑结构和资源映射机制，不同点在于：BiChord 采用传统贪婪式搜索算法；BiChord 的路由表维护算法是：节点每分钟检查路由表的每个入口，当某个路由表的入口中包含的有效路由指针数目少于 2 时，则启动主动探测算法查找该入口对应的键值空间的中间位置的节点，在该目标节点的路由表中随机选取至多 4 个属于该键值空间的节点返回给主动探测节点。实验时节点总数 *N* 为 2 000，节点负载采用单位时间(T=1 min)内处理的消息数量来衡量，每个节点每分钟发送一条查询消息。物理网络采用 Transit-stub 模型，路由器数量为 5 050，物理链路 25 168 条，链路平均延迟为 40.5 ms，每个节点随机连在一个路由器上。

实验结果如图 2 所示。在 BiChord 中，节点最大负载为每分钟处理 161 条消息，最小负载为每分钟处理 21 条消息，94.5%的节点负载在每分钟处理 60 条消息以下，97%的节点负载在每分钟处理 70 条消息以下，99%在每分钟处理 84 条

(下转第 40 页)