高技术通讯 2009 年 第 19 卷 第 3 期:230 ~ 236

doi:10.3772/j.issn.1002-0470.2009.03.003

一种适合异构 P2P 网络的树形结构覆盖层①

杨 亚② 宋俊德

(北京邮电大学电子工程学院,北京 100876)

摘 要 为实现异构网络下的 P2P 应用,分析了具有异构融合特征的 P2P 网络的特点,提出了与之相适应的基于二叉树结构的覆盖层网络(TSOHEN)的设计方法。该方法根据节点的不同功能和属性将节点分为普通节点和混合节点两大类,并为每类节点设计了适应异构特征的路由表,对各类节点的加入和离开功能设计了相应的算法,并通过混合节点实现跨网的 P2P 查询操作。数值和仿真结果表明,该覆盖层设计能够有效地适应异构网络的环境,树形结构也没有使得根节点和叶节点的负荷产生明显的区别,各混合节点的负载也基本平衡。在大规模节点数量的情况下,TSOHEN 的各种算法仍具有良好的收敛性。

关键词 对等网络 P2P, 覆盖层网络, 异构网络

0 引 言

覆盖层网络(overlay network)一直是 P2P 技术的研究重点之一。目前已实现的有基于分布式哈希表(DHT)的 Chord[1]、Tapestry[2]、Pastry[3] 和 CAN[4] 等。这些网络在节点的加入、离开、查找和路由表的更新方面,有各自的优点。但它们多为基于单一网络的情景,一般都默认在其底层都是采用 IP 的机制。在覆盖层看来,所有的节点都具有 IP,都是完全对等的实体。但在异构的两个网络中,底层的实现机制未必都是采用 IP,而有可能会是非 IP 的结构,例如一个是无线局域网(WLAN),另外一个是蓝牙网络。这样在覆盖层就不能简单地将不同网络的节点看作是完全对等的 peer,而需要考虑其属于不同网络的特性。本文主要解决的问题是,在异构网络情况下如何实现 P2P 的应用。

本文首先分析了具有异构融合特征的对等网络的基本特点,将其进行了抽象并建模。在此基础上,提出了在异构网络融合的条件下,基于二叉树结构的覆盖层网络(tree structure overlay for heterogeneous network,TSOHEN)的设计思路和算法。TSOHEN 根据节点的特性将其划分为普通节点和混合节点两类,为两类节点设计了新的路由表。为了实现两个不同子网中的节点能够互访,对混合节点的操作进行了重点研究,对混合节点的加入、退出和查询等关键操作设计了新的算法,并给出了各种操作的示例流程。论文后半部分对算法的开销进行了详细的分析,并在仿真实验平台上,对这些算法进行了验证,并和其它算法进行了比较分析。

1 异构网络的特点分析

一个典型的异构网络的节点分布和网络覆盖如图 1 所示。

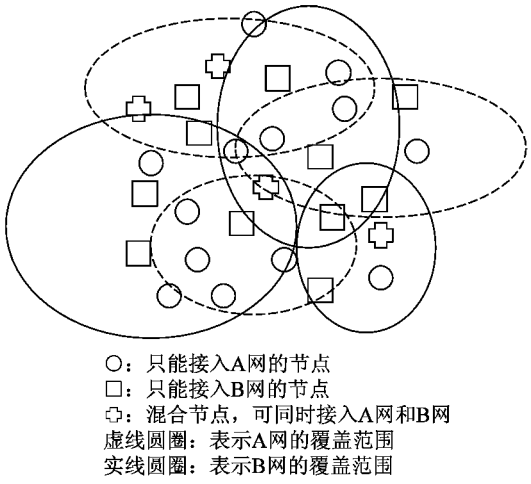

图 1 异构网络的节点分布和网络覆盖

以上三种节点,从逻辑上可分为两类:一类是只具有一种接入能力的节点,称之为普通节点;另外一

① 863 计划(2006AA01Z206)资助项目。
② 男,1975 年生,博士生,讲师;研究方向:数据挖掘,计算机网络;联系人,E-mail:yangya@bupt.edu.cn
(收稿日期:2008-01-30)

种是具有两种接入能力的节点,即混合节点。

图 1 经过一定的变换,可成为如图 2 所示的逻辑上的分布和覆盖图。

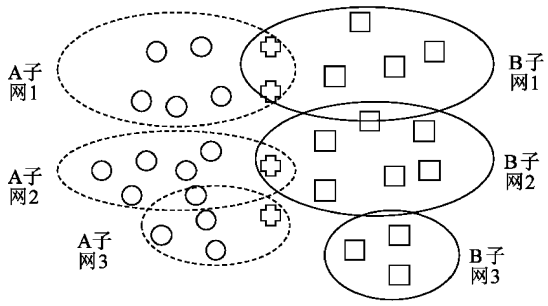

图 2　逻辑上的异构网络的节点分布和网络覆盖

由于 A 网和 B 网是两种结构不同的网络,它们的实现机制和底层协议都有区别,即便是物理上有重叠区域,但仍然不能互相通信。但如果有混合节点出现在两个网络的重叠处,那么 A 网和 B 网,就能够通过混合节点实现两种网络协议的转换,实现 A 网和 B 网的互相通信。显然,混合节点是连接 A 网和 B 网的桥梁,它的负载应该也是较高的。混合节点的出现,是在单一一种网络下遇不到的特殊情况,也是本文分析研究的重点。

2　适应异构网络的覆盖层设计

在以往大多数覆盖层机制研究中,采用树形结构的较少。一个很重要的理由就是担心根节点和叶节点的负载不均衡,根节点的负载过高。但树形结构理论完备,结构简单,在实际应用(如数据库系统)中也都有良好的表现。BATON[5]和 BATON*[6]提出了以树形结构作为覆盖层网络的基本结构,它能够有效地解决以上提到的根节点和叶节点负载不平衡的问题。国内的相关研究(如 MPPBTree[7])也采用了平衡树的结构,使被查询的内容达到一种近似平衡的分布特征。

参考以上树形结构覆盖层的设计理念,结合异构网络的特点,本文将混合节点作为树根,其左右两个子树分别对应两个不同的子网 A 和 B。由于 A/B 两个子网的节点数量可能相差很大,整个树未必是一个平衡二叉树,但根节点以下的左右两个子树内部,仍然保持成为一个平衡二叉树。按照这种设计方法,可以将图 2 所示的结构,变换成为如图 3 所示的树形结构。

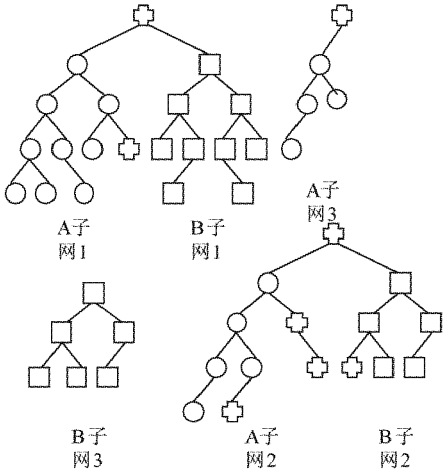

图 3　树形结构的覆盖层

在 BATON 设计的覆盖层中,可以通过横向的邻居节点的链接,方便地访问到同层的其它节点。但在异构网络的情况下,这种访问只能在同一个子树内进行。如果需要查找对侧子树的节点(例如在本侧子树中查不到所需的信息时),由于左右子树之间的节点不可直接访问到,所以也没有办法获取和保存同一层中对侧子树的节点信息。如何从一侧的节点出发,访问到另外一侧中的节点,则是需要解决的一个问题。

对于这些跨越左右子树的访问,可以全部由根节点来完成。因为它能够感知左右两侧子树中的所有节点。但其缺点也很明显,即根节点的负载很重。

如果一棵树中有多个混合节点(假设混合节点数占全部节点数的 10%),那么,把这些混合节点放到哪里,才能够更好地实现负载分担和提高性能的目的呢?

本文考虑把这些混合节点放到不同的层上去。比较理想的情况是每一层都至少有一个混合节点。这样,左子树中的节点要访问右子树的节点,只需要先访问到同层的混合节点,就能够通过该混合节点路由到右子树中的节点,而不需要都经过根节点了。这样既减轻了根节点的负担又能够加快路由的速度。

假设每个节点查询对侧子树的机会是均等的,节点越多的层,其发出的查询对侧的请求也就越多。这样应该部署更多的混合节点在节点比较多的层上,以便进一步分担负载。

3　路由表的构造

由于加入了混合节点,本覆盖层网络的路由表

的构造与 BATON 有所不同。以图 4 中的树形结构为例,说明各种节点的路由表的构造。

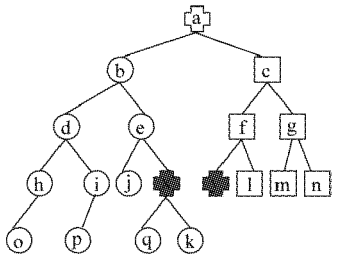

图 4 一棵典型的具备混合节点的异构网络二叉树

3.1 根节点的路由表

根节点肯定是一个混合节点。其路由如表 1 所示。

表 1 根节点的路由表

节点:a 层数=0 编号=1 后备混合节点=x
左孩子=b 右孩子=c 左邻接点=k 右邻接点=y

左子树 混合节点总数=1

层数	普通节点数	混合节点数	最右边的节点	是否混合节点
1	1	0	b	否
2	2	0	e	否
3	3	1	x	是
4	4	0	k	否

右子树 混合节点总数=1

层数	普通节点数	混合节点数	最左边的节点	是否混合节点
1	1	0	c	否
2	2	0	f	否
3	3	1	y	否

在根节点的路由表中增加了一个后备混合节点,目的是当根节点退出时,可以很快地找到后备根节点来替代它,以增强系统的健壮性。另外,在左子树表中记录了左子树每一层最右边的节点信息(右子树类似),在后面的节点加入和退出时将会用到。

3.2 混合节点的路由表

以混合节点 x 为例,路由表如表 2 所示。

表 2 混合节点的路由表

节点:x 层数=3 编号=4 根节点=a 是否混合节点=是
父亲=e 左孩子=q 右孩子=k 左邻接点=q 右邻接点=k
本层最右边的混合节点=x

左路由表

距离	节点	左孩子	右孩子	下界	上界	是否混合
1	j	null	null			否
2	i	p	null			否

右路由表

距离	节点	左孩子	右孩子	下界	上界	是否混合

对侧路由表

距离	节点	左孩子	右孩子	下界	上界	是否混合
0	l	null	null			否
1	m	null	null			否
2	n	null	null			否

与 BATON 相比,除了在左右路由表中增加“是否混合”的标志以外,还增加了一个对侧路由表,用于记录对侧子树中同层节点的信息。如果该混合节点是该层最右边的混合节点,则还需要增加一个记录本层混合节点的表,以便在该节点退出时,很快找到一个替代节点。另外,也可以在对侧路由表更新时,通知这些混合节点同步更新其对侧路由表。

3.3 普通节点的路由表

以普通节点 h 为例(见表 3)。

表 3 普通节点的路由表

节点:h 层数=3 编号=1 根节点=a 是否混合节点=否
父亲=d 左孩子=o 右孩子=null 左邻接点=o 右邻接点=d
本层最右边的混合节点=x

左路由表

距离	节点	左孩子	右孩子	下界	上界	是否混合

右路由表

距离	节点	左孩子	右孩子	下界	上界	是否混合
1	i	p	null			否
2	j	null	null			否

每个普通节点都按照 BATON 的规则保存距离它 2^i 的节点,如果该位置上是混合节点,则在“是否混合”一列中加以标记。另外,还增加了一项“本层最右边的混合节点”,这样保证在左右路由表中都没有的混合节点时,至少可以通过本层最右边的混合节点,将查询消息转发到对侧的子树中去。

4 节点的加入、退出和查找算法

4.1 节点加入算法

节点可分为普通节点和混合节点 2 类。由于普通节点只具备一种接入能力,只能找到左子树或者右子树中的某个节点。在找到后,按照 BATON 中的加入方法,先确定其父节点,然后完成实际的加入操作。

混合节点比较特殊,它能够同时联络到左右子树中的节点,首先应确定把它加入到左子树还是右子树中。考虑到左右子树的节点数量可能差距较大,可以按照各自节点数占整棵树节点总数的比例进行分配。

下面是加入到左子树的例子:

```
JoinLeftSubTree(Node n)
int L = JudgeLevel();
Node a = MostRightNodeOfLevel(L);
```

```
IF ( NumberOf(a) < 2^(L-1)) THEN
{JoinAsRightChild(MostRightNodeOfLevel(L-1));
  Node y = root.RightTable[L];
  IF(y! = NULL) THEN
    BuildOtherSideTable(GetRightTableFrom(y))
  ELSE
  {  Node z = NearestParentHybridNode(L-1);
    IF z ! = NULL THEN
        BuildOtherSideTable ( GetOtherSideTable
        (z))
    ELSE
    { p = root.RightTable[MaxLevel];
        BuildOtherSideTable ( GetRightTableFrom
        (p);
    }}}
    ELSE Replace(n, a)
```

4.2　节点的退出算法

• 情况 1,普通节点退出:普通节点的退出方法和 BATON 基本一样,但需要增加通知根节点的操作,以便根节点更新其路由表。

• 情况 2,根节点退出:如果根节点是整棵树中的最后一个混合节点,它的退出将意味着左右子树所代表的两个子网无法互相访问到,进而形成两个单独的网络。直到有新的混合节点出现时,才能将两个网络重新合并起来。如果有其它的备用混合节点,则可以将备用混合节点调整到根节点的位置(使其先离开网络,参见混合节点的退出机制),并通知所有的节点,根节点发生了变更,复杂度同样是 $O(N)$。

• 情况 3:将要离开的混合节点是该层最右边的节点,同时该层还有其它混合节点。这时可以从剩余的混合节点挑选一个取代它,然后通知根节点,告知其该层最右边的节点发生了变化。如果退出的混合节点是本层中最后一个混合节点,则该混合节点的退出,将造成本层的左右两个子树间无法通过该混合节点进行通信,必须通过该层的上一层或下一层的混合节点才能完成跨越。方法之一是从其它层找一个混合节点来代替。但这种方法将引起其它层上的节点对其路由表也必须进行调整,开销较大。

• 情况 4,将要离开的混合节点不是该层最右边的节点(对于右子树则正好相反,下同)。这时可将其看作是普通节点的离开,除了向邻居节点发出路由表更新的消息,还需要给同层的最右边的混合节点发出更新消息。最后还需要向根节点报告。

4.3　查找算法

由于存在两个不同的网络,不同网络的节点查找对方时,都必须通过混合节点进行消息的转发,这样势必给混合节点增大负荷。所以在节点发起查找请求后,首先应在该节点所在的子网内进行。只有在本子网内查找不到时,才将查询请求经过混合节点转发到另一个子网中去。

节点接收到查询请求时可分为以下几种情况:

• 要查找的值不在本节点负责的范围之内:将查询的请求转发给本子树中的其它节点(邻居节点、孩子节点或邻接点)。

• 要查找的值在本节点负责的范围之内,而且要找的值保存在本节点中:直接返回结果。

• 要查找的值在本节点负责的范围之内,但本节点并没有保存:这表示在本侧子树内的所有节点,都不会有该值存在,但有可能在对侧的子树中,需要把查询请求转发到对侧中去。通过在路由表中保存的混合节点或者根节点的信息,只需要一步就能够将查询请求转发给它们。

```
SearchOnOtherSide(Node n,Key:V)
IF NodeIsRoot(x) THEN
{ IF NodeIsInLeftSubTree(n)
  THEN
    { FOR(i=0;i<x.RightTableLenth();i++)
    {IF(x.RightTable[i] <= V < x.RightTable
    [i]) THEN Send (x.RightTable[i], "Exact-
    Search",n,V);}
    j=Random(1, x.RightTableLenth());
    Send (x.RightTable[j], "ExactSearch", n,
    V);}}
  ELSE //节点 n 在右子树,需要查询左子树,
  略}
ELSE
    IF NodeIsInLeftSubTree(n) THEN
    { Node m=FindFarthestInRightTable(lower <
    =V);
    IF m ! = NULL THEN
      Send(m,"ExactSearch",n,V)
    ELSE Send(root,"SearchOnOtherSide",n,V)
    }
  ELSE//节点 n 在右子树,略
}//End of SearchOnOtherSide
```

查找过程举例(图 5):通过非根节点的混合节点进行转发的例子:从节点 o 开始,查找 74。74 在

节点 g 上保存。

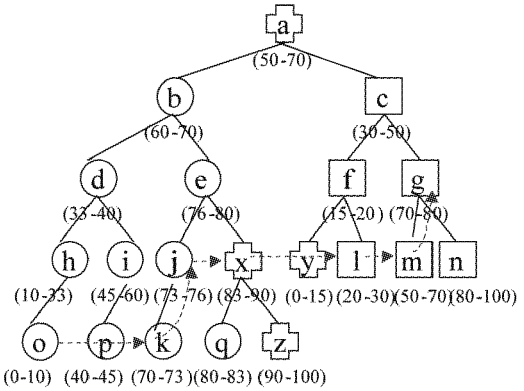

图 5 通过同层混合节点将查询请求转发给对侧子树

首先,在节点 o 所在的左子树中发起一个本子网的查找请求,按照算法,将经历 o—k—j 的查找路径。虽然节点 j 负责 73 – 76 的范围,但在节点 j 上没有保存目标结果。这表示在整个左子树内,都没有这个目标,只能搜索右子树中是否有该目标存在。这时将从 j 开始发起一个查询对侧子树的查询请求。

节点 j 通过其右路由表查找是否有同层的混合节点存在,如果有,则将查询请求发给同层的混合节点,然后通过该混合节点把查询请求转发至右子树。其路由表中有一个混合节点 x,x 收到查询请求后,将检查其右路由表,找到满足条件(下界小于 73)的最远的一个节点 l,然后根据算法,l 再转发给 m,m 转发给 g。最后在节点 g 上找到所需的目标。

5 算法复杂度分析

5.1 节点加入算法的复杂度分析

和 BATON 相比,普通节点加入时仅仅多了一条给根节点发送的加入消息,复杂度仍为 $O(\mathrm{Log}N)$。

混合节点加入时,主要开销在 JoinLeftSubTree(Node n),在第一步中确定加入到哪一层,其复杂度显然为 Log(N),即子树的高度,用来获得每一层目前的节点数。在第二步加入子树的算法中,JoinAsRightChild(MostRightNodeOfLevel($L-1$),可以从根节点的左子树表中很方便地找到第 $L-1$ 层的最右边的节点 x(我们以混合节点 z 加入作为节点 x 的孩子为例),节点 x 需要向其路由表中同层的所有邻居节点发出更新消息,告知其右孩子 z 的加入。这里最多需要向 $2L_1$ 个邻居节点发送消息(L_1 是 x 所在的层数)。

这些邻居节点,将发送最多 $2L_2$(L_2 是 z 所在的层数)条消息给它们的孩子节点,告知这些孩子节点有与之同层的新节点加入,这些孩子节点还将返回 $2L_2$ 条消息给 z,以便 z 根据这些信息来构造路由表。z 需要发送一条消息给它的邻接点 a,以便其更新路由表。另外 z 需要给根节点 a 发送一条消息,告知新的混合节点的加入。这样最多总共需要 $2L_1+2L_2+2L_2+2$ 条消息 $<6\mathrm{Log}N$。

混合节点加入的复杂度仍保持为 $O(\mathrm{Log}N)$。

5.2 节点退出算法的复杂度分析

普通节点:和 BATON 的开销基本一样,但需要通知根节点,以便根节点更新其路由表。普通节点离开时所需要发出的消息数是 $2L_1+2L_2+3<4\log N$(L_2 是离开节点所在层的层号,L_1 是其父亲节点的层号)。

根节点:根节点退出时,需要向左右两边子树的所有节点发出更新消息,通知它们根节点的离开,更新其路由表。这个复杂度是 $O(N)$。

混合节点:假设混合节点占节点总数的 10%,则第 L 层有 $2^L\times 10\%$个混合节点。

情况 1:将要离开的混合节点不是该层最右边的节点(对于左子树而言,右子树则正好相反,下同),而且同层上还有可用的其它混合节点,这种概率为 $\frac{2^L\times 10\%-1}{2^L\times 10\%}$。这时可将其看作是普通节点的离开,除了向邻居节点发出路由表更新的消息,还需要给同层的最右边的混合节点发出更新消息。最后还需要向根节点报告。混合节点离开时,引起路由表更新的消息数比普通节点多 1(为 $2L_1+2L_2+4=4L_2+2$)。

情况 2:将要离开的混合节点是该层最右边的节点,同时该层还有可用的其它混合节点。其概率是 $\frac{1}{2^L\times 10\%}$。这时可以从剩余的混合节点挑选 1 个取代其位置,然后通知根节点,告知其该层最右边的节点发生了变化。开销为混合节点一次退出和一次加入的开销之和(为 $2L_1+2L_2+4+2L_1+4L_2+2=10L_2+3$)。

5.3 查找算法的复杂度分析

如果查询的目标节点和发起查询请求节点,属于同一个网络类型,则查询只需要在同一侧的子网内就能完成,其复杂度为 $O(\mathrm{Log}N)$,N 是该子网节点总数。

如果查询的目标节点和发起查询请求节点,分

属两个网络,则开销为 3 个部分构成:

(1) 在本子网内查询的开销:$O(\mathrm{Log}A)$,A 为发起查询请求的节点所在子网的节点总数。

(2) 在本子网查询不到,需要对侧网络。由于在每个节点的路由表中,都保存了同层混合节点或者根节点的信息,只需要一步就能够将查询消息转发给混合节点,再由混合节点进行转发。

(3) 在对侧子网内查询的开销:$O(\mathrm{Log}B)$,B 为目标节点所在子网的节点总数。

因此,总的查找开销是 $O(\mathrm{Log}A + \mathrm{Log}B)$。

6　实验和仿真

参考 BATON 的仿真测试环境,以及 Planetlab[8] 对 P2P 应用的测试要求,本文设计了仿真实验环境,并主要采用 NS2 进行了仿真。实验中对不同的网络规模进行了测试,总节点数 N 从 1000 到 10000 个。另外,我们设定混合节点占整个节点总数的 10%。

如果网络有 N 个节点,范围为[1, 1000000000]的 $1000 \times N$ 个数据,将被批量地插入到这 N 个节点中。每次测试,都将执行 1000 次精确查询,然后取平均开销值。为了模拟出各个事件(节点加入或离开)是按照不同的序列发生,实验执行了 10 次,每次都采用了不同的序列,最后得出的是这 10 次实验的平均值。

6.1　节点加入和离开时的消息数

和 BATON 相比,节点加入和离开时的消息数略高,因为每个节点加入或离开时都要向根节点汇报。节点加入和离开的消息数见图 6。

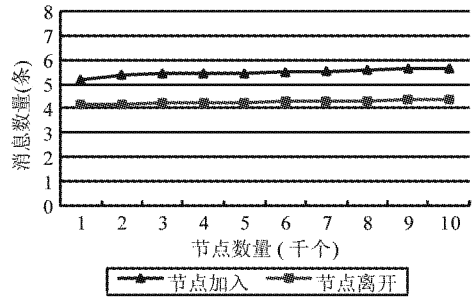

图 6　节点加入和离开时的消息数

6.2　加入离开时,更新路由表的消息数

对于普通节点,只能加入左子树或者右子树。对于混合节点,按照算法,必须先确定好加入哪一侧的子树后,再完成具体的加入操作。对于普通节点和混合节点,它们加入时的开销基本相等。退出时,由于混合节点承载了更多的信息,其离开时的开销比普通节点稍大(图 7)。

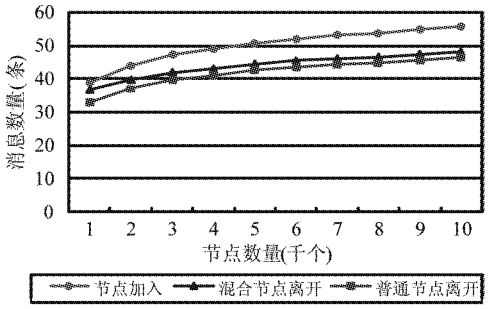

图 7　节点加入和离开时更新路由表的消息数

6.3　数据插入和删除的开销

由于在数据插入和删除时,只需要在一侧的子树内进行,所以本算法比 BATON 的开销稍小(图 8)。

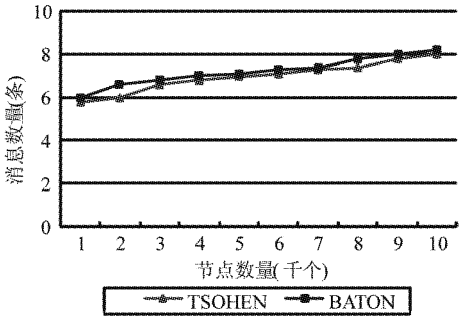

图 8　数据插入和删除的消息数

6.4　数据精确查询的开销

假设数据插入时是在两侧的子树均衡的,则在进行数据的精确查询时,有 50% 的可能性在本侧子树,50% 的可能在对侧子树中(图 9)。

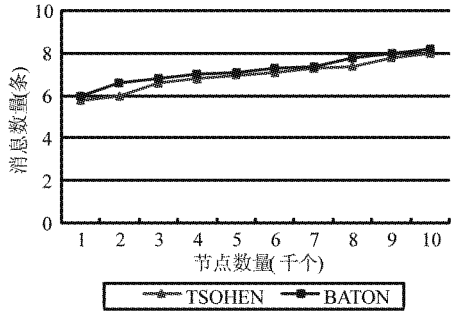

图 9　精确查询的消息数

TSOHEN 的开销和 BATON 相比略高，主要原因是由于如果所查找的资源不在本侧子树时，需要将查询消息转发至对侧，并在对侧子树中重新发起查询。

从以上实验结果可以看出，以上各种操作的开销，并不因为节点数的线性增长而同样发生线性的增长，基本保持为 $O(\mathrm{Log}N)$。说明算法在节点数较大时，仍具有较好的收敛性。

7 结 论

在异构网络中实现 P2P 应用，为覆盖层的构造带来了新的挑战。混合节点一方面可以当作普通节点实现资源的共享，另一方面也充当了两个网络之间连接的桥梁。TSOHEN 重点针对混合节点，提出了新的节点加入、退出和查找的算法，最后在测试平台上对算法和其性能进行了验证。数值和仿真结果表明，TSOHEN 能够有效地适应异构网络的环境，树形结构也没有使得根节点和叶节点的负荷产生明显的区别，各混合节点的负载也基本平衡。在大规模节点数量的情况下，TSOHEN 的各种算法仍具有良好的收敛性。

参考文献

[1] Karger D, Kaashoek F, Stoica I, et al. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 ACM SIGCOMM Conference, San Diego, CA, USA, 2001. 149-160

[2] Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of the 18th IFIP/ACM International Conference of Distributed Systems Platforms, Heidelberg, Germany, 2001. 329-350

[3] Zhao B Y, Kubiatowicz J D, Joseph A D. Tapestry: an infrastructure for fault-tolerant wide-area location and routing: [Technical Report CSD-01-1141]. Berkeley: University of California-Berkeley, CA, USA, 2001

[4] Ratnasamy S, Francis P, Handley M, et al. A scalable contentaddressable network. In: Proceedings of the 2001 ACM Annual Conference of the Special Interest Group on Data Communication, San Diego, CA, USA, 2001. 161-172

[5] Jagadish H V, Ooi B C, Vu Q H. Baton: a balanced tree structure for peer-to-peer networks. In: Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005. 661-672

[6] Jagadish H V, Ooi B C, Vu Q H, et al. Speeding up search in peer to peer networks with a multiway tree structure. In: Proceedings of the 2006 ACM SIGMOD Conference, Chicago, Illinois, USA, 2006. 1-12,

[7] 许立波，于 坤，吴国新. 基于匹配路径和概率平衡树的 P2P 语义路由模型. 软件学报，2006，17(10)：2106-2117

[8] Chun B, Culler D, Roscoe T, et al. Planetlab: an overlay testbed for broad-coverage services. In: Proceedings of the ACM SIGCOMM Computer Communication Review, Karlsruhe, Germany, 2003. 33(3)

TSOHEN: a tree structure overlay for heterogeneous P2P networks

Yang Ya, Song Junde

(School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract

In order to apply P2P to heterogeneous networks, the paper proposes a new design method based on the tree structure overlay for heterogeneous networks (TSOHEN). The method classifies the nodes into two types according to their functions and attributes, the common nodes and the hybrid nodes, and designs a new routing table for each type of the nodes and new algorithms for the joining and leaving of the nodes. Through hybrid nodes, it implements the P2P searching function across different networks. The simulation result indicates that the overlay design works well for heterogeneous networks. The tree structure does not cause obvious difference for loading between root node and leaf nodes. And the loading of hybrid nodes is almost balanced. The algorithms of TSOHEN show good convergence even in the large-scale situation.

Key words: peer-to-peer networks (P2P), overlay network, heterogeneous network