层次式 Chord:物理拓扑感知的结构化对等网

肖卓程　荆金华

(复旦大学计算机与信息技术系　上海 200433)

摘　要　本文针对对等网由于逻辑网络和物理网络的拓扑结构不匹配导致物理路由效率低下的问题,在结构化 P2P 网络 Chord 的基础上,提出一种层次式 Chord 模型。模拟实验表明,该模型能够有效提高物理路由的效率,并保持良好的逻辑路由效率和较低的维护代价。

关键词　结构化对等网,分布式散列表,物理网络,逻辑网络,Chord,层次式 Chord

Layered Chord: Physical Topology Awared Structured P2P Network

XIAO Zhuo-Cheng　JING Jin-Hua

(Department of Computer Information and Technology, Fudan University, Shanghai 200433)

Abstract　This paper analyzed the unmatched topology problem between the overlay network and the physical network, which result in inefficient routing. Based on the structured P2P network Chord, this paper proposed a layered Chord model. Simulation experiment shows that layered Chord can greatly improve routing efficiency so as to reduce physical path length of lookup and keep close overlay path length of lookup and network maintenance against Chord.

Keywords　Structured P2P network, DHT, Physical network, Overlay network, Chord, Layered chord

1　引言

P2P 网络是在应用层上构建的一种逻辑网络,由于没有考虑物理网络的相关信息。逻辑网络和物理网络的拓扑结构不可避免地出现不匹配。在逻辑网络中,相邻的节点,常常在物理网络中相距甚远,因此逻辑网络中的一次逻辑路由可能会经过物理网络中的多次物理路由。这种低下的物理路由效率导致了 P2P 网络中查找的物理路径较长的问题,并增大网络的通信代价。

P2P 网络根据资源查找的方式主要可以分为 3 种类型:早期的集中式 P2P 网络、中期的非结构化 P2P 网络和后期的结构化 P2P 网络。本文在结构化 P2P 网络 Chord 的基础上,提出一种层次式的 Chord 模型。该模型能够在不影响 Chord 的其他性质的前提下,提高物理路由的效率,从而有效地降低查找的物理路径长度。

2　Chord 简介

Chord 是由 MIT 提出的一种结构化 P2P 网络,它的主要应用有分布式存储、文件共享等。我们在下面对其结构和查找算法做简要介绍,更多细节请参考文[1]。

在 Chord 系统中,节点和资源都被 DHT 函数映射到一个相同的地址空间。我们可以用一个环来表示这个空间。在该空间中,节点的散列值被称为节点 ID,资源的散列值被称为键。每个节点负责一段地址空间中的键,这段地址空间表现在环中就是界于该节点(包括该节点)和该节点的前继节点之间的圆弧。节点还维护了供查找使用的路由表,这个路由表的第 k 条记录为在地址空间中和该节点的距离大等于 2^{k-1} ($1\leqslant k\leqslant m$,地址空间为 0 到 2^m-1)的最近节点。图 1 是一个地址空间为 0 到 2^6-1 的 Chord,节点 N8 负责的键的范围为 1 到 8,它的路由表中第 4 项记录为 N21,表示距离该节点大于等于 8 的最近节点为节点 N21。

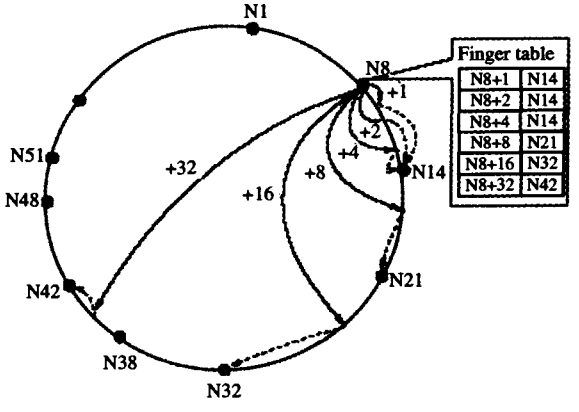

图 1　节点 8 的路由表

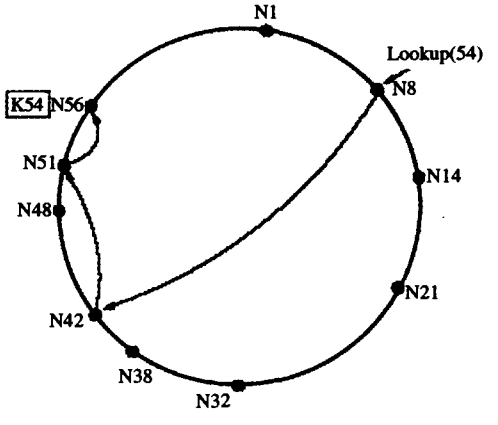

图 2　从节点 8 开始查找负责键 54 的节点

肖卓程　硕士研究生,主要研究领域为对等网计算、信息检索等。荆金华　副教授,主要研究领域为下一代计算机网络、分布式计算等。

Chord 的本质就是查找服务，即给定一个键，找到负责该键的节点。在查找的时候，节点首先判断自己是否负责该键，如果没有负责该键，则节点根据路由表中的记录，将这个查询请求转发给和该键最接近的节点，如此下去，直到最终找到负责该键的节点。如图 2 所示，节点 N8 查询负责键 54 的节点，通过 3 次转发，最终找到负责该键的节点为 N56。

3 层次式子网模型和层次式 Chord 模型

3.1 层次式子网模型

现实的物理网络的拓扑结构呈现一种层次式的特点，我们可以用层次式子网模型来表示这种拓扑结构[4]。在该模型中，存在多个不同级别的子网，高一级子网由低一级子网组成；最低级别的子网称为一级子网，最高级别的子网称为顶级子网；级别越低的子网内的节点间的物理距离越短。

图 3 为某大学的校园网拓扑，它是一个具有两级子网层次的网络。南区子网和北区子网为一级子网，而大学子网为二级子网。其中节点 S1、S2、S3 属于南区子网，节点 N1、N2、N3 属于北区子网，南区子网和北区子网构成大学子网。一级子网内的节点间物理距离较短，而一级子网间的节点间的物理距离较长。例如节点 S1 和 S2、S3 的物理距离较短，而 S1 和 N1、N2、N3 的物理距离较长。

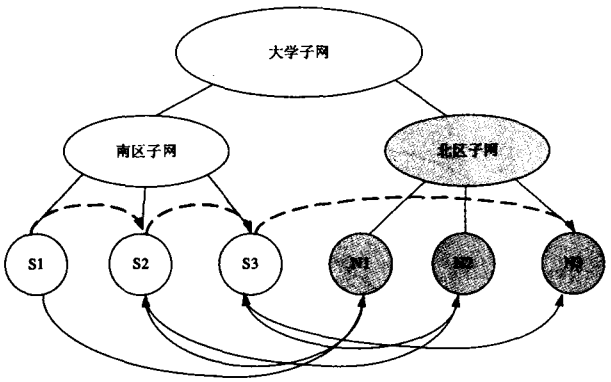

图 3 某大学的校园网拓扑

3.2 层次式 Chord 模型

文[4]提出一种层次式的 Chord 模型，其定义如下：

1)每个子网中的所有节点形成一个子网 Chord，下级子网 Chord 包含的节点是上级子网 Chord 包含的节点的子集；

2)每个节点加入自己所属的所有子网对应的子网 Chord，并且节点在其所属的所有子网 Chord 中具有相同的节点 ID；

3)只有顶级子网 Chord 才提供真正的键的存储，其他层次的子网 Chord 仅用于辅助路由。

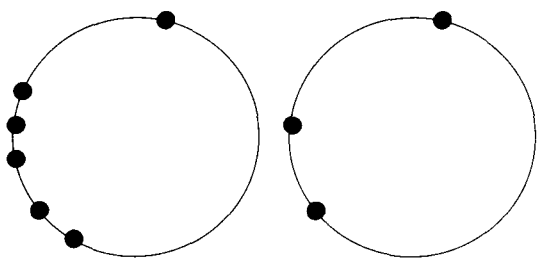

图 4 大学子网 Chord 和南区子网 Chord

以上面提到的大学校园网为例，图 4 为该大学校园网的大学子网 Chord 和南区子网 Chord 的一个实例。可以看到，南区子网 Chord 包含的节点是大学子网 Chord 包含的节点的子集，并且相同的节点（如节点 S3）在其所属的南区子网 Chord 和大学子网 Chord 中的节点 ID 都相同（体现在图中是该节点在大学子网 Chord 和南区子网 Chord 中的位置都相同）。

4 层次式 Chord 的设计与实现

基于 Chord，我们增加了子网识别模块、层次式构造和维护模块以及层次式的查找模块，如图 5 所示。

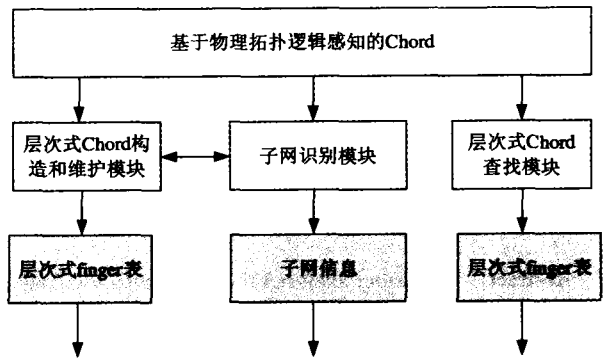

图 5 层次式 Chord 的总体框架

4.1 子网信息管理模块

如图 5 所示，层次式 Chord 的构造和维护模块依赖于子网识别模块。每个加入层次式 Chord 的节点都必须获得自己所属的子网的相关信息，这样才可以进行相应子网 Chord 的构造和维护操作。

层次式子网识别模块可以采用集中式的方式来实现，但这种方法需要专门设置的主机，并对这些主机产生较大的负载，同时影响网络的可扩展性。因此，我们在这里采用一种基于 Chord 的分布式的子网识别方法，它主要由子网信息存储模块和子网识别模块组成。

4.1.1 子网信息存储模块

1)子网信息由该子网的下一级子网的 ID 和测量节点的 IP 地址构成。

2)将子网信息以子网的 ID 为键，作为一般的文件分布式的存储在顶级子网 Chord 中。

可以看到，采用这种基于 Chord 的存储方式，子网信息分布式的存储到网络中的节点中，从而使由于查询和存储产生的负载分布到多个节点上。

4.1.2 子网识别模块

1)每个节点在加入层次式 Chord 时，根据默认的顶级子网 ID，获得顶级子网对应的子网信息。

2)从获得的子网信息中抽取出下一级子网的 ID 以及测量节点，分别和这些下一级子网的测量节点进行网络延迟测量（例如进行 RTT 测量）。

3)如果节点和某个下一级子网的测量节点的测量值符合该级子网的延迟范围，则该节点可以确认其属于该子网，转 4)。如果节点和所有子网的测量值都不符合该级子网的延迟范围，则可以认为该节点不属于任何一个下一级子网，转 5)。

4)如果到达最低子网，则识别算法结束，否则节点以该子网的 ID 为键，从顶级子网 Chord 中获得该子网的相关信息，转 2)。

5)如果到达最低子网,则识别算法结束,否则节点自己产生一个新的子网,子网 ID 为该节点的 ID,并设置自己为该子网的测量节点,然后将子网信息以子网 ID 为键发布到顶级子网 Chord 中。

从上面的算法可以看出,采用这种方法,将子网识别的工作是分布式进行的。节点在进行子网识别的过程中,将从多个节点中获得相应级别的子网信息,并和多个节点进行网络延迟测量,从而获得自己在所有级别上所属的子网信息。

4.1.3 子网信息的维护

子网信息的维护主要是对子网的测量节点的信息进行维护。子网中的节点将定期查询自己所属子网的测量节点是否有效。如果失效的话,节点将充当该子网的测量节点,并将相应的信息更新到顶级 Chord 中,以便后来的节点能够继续加入该子网。此外为了保持稳定性,我们可以对每个子网设置多个测量节点。

4.2 层次式 Chord 的构造和维护模块

节点在获得自己在每个层次上所属的子网的相关信息后便可以在每个层次上独立地进行子网 Chord 的构造和维护操作。即节点将在每个层次上调用 Chord 的节点加入方法加入其所属的子网 Chord,并在该层次上调用 Chord 的节点维护方法维护该子网 Chord。

4.3 层次式 Chord 的查找算法

每个节点在查找存储某个键的节点的时候,首先从其所属的最低级别子网 Chord 开始查找,如果在该级子网 Chord 中找不到存储该键的节点,则在该子网的父子网 Chord 中继续查找,如此下去,直到顶级子网 Chord。

4.4 层次式 finger 表

Chord 的构造和维护以及查找都以 finger 表为基础,因此这种层次式的 Chord 理论上需要节点在每个层次上维护一个独立的 finger 表。因此对于 k 层的层次式 Chord 来说,每个节点需要维护 k 倍于 Chord 的 finger 表。我们在下面将采用一种以压缩方式存储的层次式 finger 表来代替上面的 k 个 finger 表,从而大大降低 finger 表所需要的空间。

4.4.1 层次式 finger 表的定义

层次式 finger 表相对于原来每层独立存储一个 finger 表的方法,采用了混合存储的方法,即节点将所有层次的 finger 表存储在一起,并在横向和纵向上分别进行了压缩,减少 finger 表所需要的空间。

表 1 为某个层次式 Chord 节点没有压缩过的 finger 表,设层数为 2,finger 表长度为 5。

表 1 未压缩的 finger 表

	一级 finger 表	二级 finger 表
1	4	5
2	5	5
3	5	5
4	10	18
5	18	18

如表 1 所示右边标出的二级子网的三个'5',因为 Chord 的查找过程是自底向上的,如果最下面的'5'不满足查找要求,则上面的两个'5'也将不满足,可见上面的两个'5'的存在是浪费空间的。因此我们将上面的三个'5'存储为一个'5',即纵向压缩。再如表 1 标出的一级子网的第 2 项到第 5 项,由于在层次式 Chord 的查找过程中,当二级子网的 finger[1]不满足查找要求时,将向一级子网进行路由,此时将查找比 finger[1]小,且最接近 finger[1]的表项。如表 1 二级子网的 finger[1]为'5',则一级子网中只有'4'满足要求,可见一级子网中的其余表项也是浪费空间的。因此我们将一级子网中的第 2 项到第 5 项除去。同时,为了区分不同级别子网的 finger 表项,我们在每个表项后关联一个层次属性,经过压缩后得到的层次式 finger 表如表 2 所示,我们可以用链表来表示这个结构。

表 2 压缩后的 finger 表

层次式 finger 表
4
5(2)
18(2)

可以看到,经过压缩后的 finger 表能够有效地减少 finger 表的大小,并且实验证明层次的多少对压缩后的 finger 表的体积影响不大。

4.4.2 层次式 finger 表的构造和维护

由于采用了新的 finger 表结构,因此需要对原来的 Chord 的 finger 表的访问原语进行一定的修改。经过分析,Chord 对 finger 表的访问原语可以归纳如下:

1)在 Chord 的构造和维护过程中,设置第 i 层的第 j 个表项为某个节点 ID,即 set_finger(i,j,node),这里 node $\geqslant 2^{i-1}$;

2)在层次式查找过程中,读取第 i 层的第 j 个表项,即 get_finger(i,j)。

我们采用链表的方式来实现以上两个 finger 表的访问原语。链表中每个 finger 对象有一个 TTL 值,这个 TTL 值可以设置为维护周期的两倍。如果节点在两个维护周期内都没有对该 finger 对象的 TTL 值进行刷新,则表示这个 finger 对象对应的 finger 表项已失效,节点将从链表中删除该对象,并回收资源。

算法 1 finger 原语

```
set_finger(i,j,node)
 pre=null
 cur=head;
 while cur. node(node && i)=cur. layer
     pre=cur;
     cur=cur. next;
 if cur. node>node
     finger=new finger(i,node);
     pre. next=finger;
     finger. next=cur;
 if cur. node==node
   if cur. layer<i
     cur. layer=i;
   cur. refresh();
get_finger(i,j)
  cur=head;
  while cur. node (2^(i-1) && i)=cur. layer
     cur=cur. next
  if i==cur. layer
     return cur. node;
```

在我们原型系统中,以上代码根据相应的调用场景都进行了优化,达到了常数时间。但由于表达和篇幅的需要,我们在这里没有对以上代码进行优化,因此这些原语的执行时间和 finger 表的大小成正比。但注意到压缩后的 finger 表比较小,即使不对其进行优化,执行时间也是可以容忍的。

5 实验

5.1 实验设置

为了证明算法的有效性,我们采用模拟实验的方式对

Chord 和层次式 Chord 的性能进行比较。测试机器为 P4 1.8 G,512M 内存。我们用 C++ 实现了两个算法:Chord 和层次式的 Chord(LChord)。前者严格按照 Chord 的规范实现,后者在前者的基础上加入了层次式 Chord 所需的数据结构和相关算法。

模拟实验中的拓扑图由我们采用 GT-ITM 生成。该拓扑逻辑为 3 层次的网络结构,一共包含 600 个节点。其中一级子网包含两个二级子网,每个二级子网包含 5 个三级子网,每个三级子网中的平均节点数目为 60。

5.2 子网识别实验

我们通过设置每个层次的子网 RTT 参数,分别构造出每个层次的子网,然后计算每个层次的子网的平均节点数目以及节点之间的平均物理距离。从图 6 中可以看出,该网络拓扑的识别曲线呈现出层次性的特点,符合层次式拓扑逻辑的特点,这说明我们的子网识别算法是有效的。

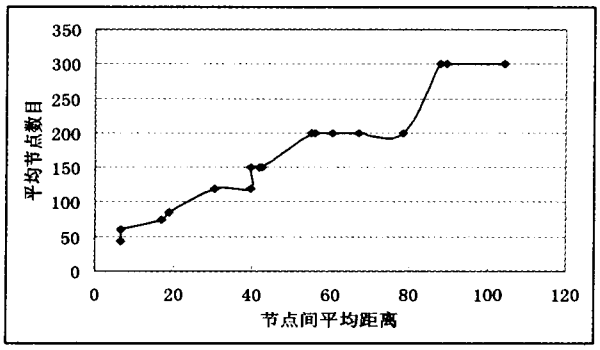

图 6 子网识别结果

5.3 路由效率及维护开销实验

我们分别构造一个层数为 3 的层次式 Chord 和一个 Chord,然后进行 100 次的随机查询和 100 次节点的随机加入,取其平均值。

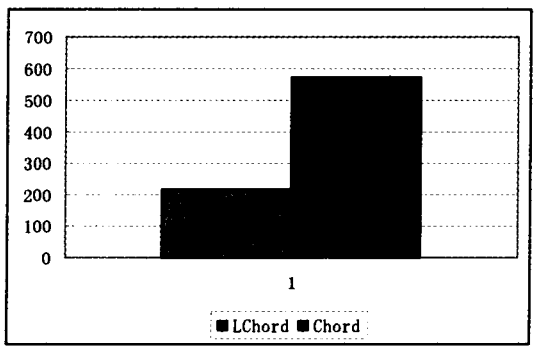

图 7 查找的物理路径长度对比

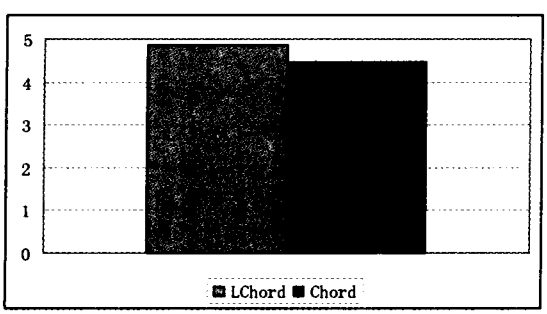

图 8 查找的逻辑路径长度对比

从图 7～图 9 中可以看出,层次式 Chord 相对于 Chord 在保持良好的逻辑路径长度的前提以及网络开销的前提下,有效地降低物理路径长度,即提高物理路由的效率。

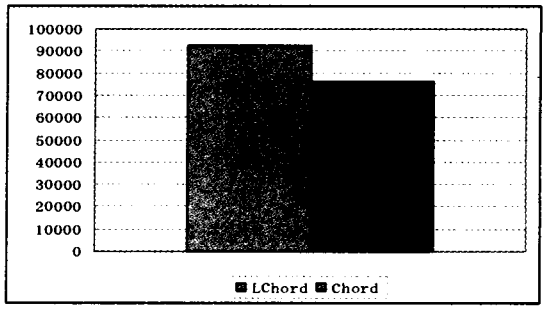

图 9 网络的维护代价对比

5.4 节点 finger 表的大小

我们分别构造层数为 1 到 10 的层次式 Chord,并计算其中的节点的 finger 表大小的平均值,如图 10 所示。从图中可以看出,采用了横向压缩和纵向压缩方式后,层次式 finger 表能有效地减少 finger 表大小,并随着层次的增大而变化不大。

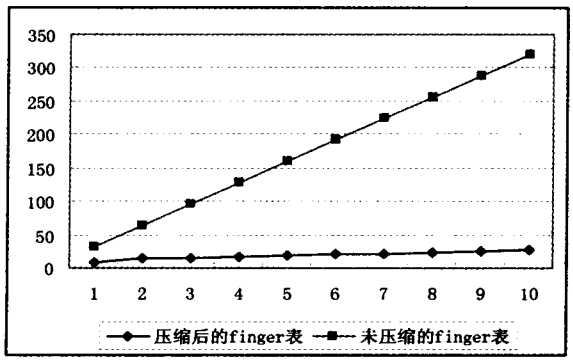

图 10 节点 finger 表大小

结论 本文针对对等网由于逻辑网络和物理网络的拓扑结构不匹配导致物理路由效率低下的问题,在结构化 P2P 网络 Chord 的基础上,提出一种层次式的 Chord 模型。模拟实验表明,层次式 Chord 能够有效地提高 Chord 的物理路由效率,并具有较低的维护开销。

参 考 文 献

1 Stoica I, Morris R, Karger D, et al. Chord: A scalable peer-to-peer lookup service for internet applications. In: Proc. of ACM SIGCOMM'01, ACM Press, 2001. 149～160

2 Waldvogel M, Rinaldi R. Efficient topology-aware overlay network. In: Proc. of the ACM SIGCOMM Computer Communication Review, 2003

3 Ratnasamy S, Handley M, Karp R M, et al. Topologically-Aware Overlay Construction and Server Selection. In: Proc. of INFOCOM, 2002

4 Freedman M J, Mazieres D, Hashing Sloppy, et al. Lecture Notes in Computer Science, 2003

5 Tsuchiya P F. The landmark hierarchy: a new hierarchy for routing in very large networks. ACM SIGCOMM Computer Communication Review, 1988, 18(4)

6 Zhao B Y, Duan Y, Huang L, et al. Brocade: Landmark Routing on Overlay Networks. In: Proc. of the IPTPS, 2002

7 Castro M, Druschel P, Hu Y C, et al. Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks. In: Proc. of the Future Directions in Distributed Computing, 2003

8 Liu Y, Zhuang Z, Xiao L, et al. A Distributed Approach to Solving Overlay Mismatching Problem. In: Proc. of the 24th Intl. Conf. on Distributed Computing Systems(ICDCS'04), 2004