一种支持负载平衡的常数度 DHT ID 分配方法

王小海,彭宇行,李东升
(国防科技大学并行与分布处理国防科技重点实验室,湖南长沙 410073)

摘 要: 能够支持负载均衡的 ID 分配方法是实现与维护 DHT overlay 的基础,已有常数度 DHT 多采用纯集中式或纯分布式方法,不能很好地解决拓扑信息维护开销与拓扑平衡程度这一矛盾.针对这一不足,在分析拓扑中通用树结构的基础上,本文提出了基于内在结构 Routing Forest 的 ID 分配方法 RFIDAM,通过规律性地聚合局部平衡信息来指导新节点的加入以实现拓扑平衡.实验表明,通过引入少量维护与路由开销,该方法使得拓扑达到节点 ID 长度差小于 2 的最优平衡目标.

关键词: P2P; DHT; ID 分配; 常数度拓扑; 负载平衡; Routing Forest
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2010) 11-2649-06

A Load Balancing-Supported Constant Degree DHT ID Assignment Method

WANG Xiao-hai, PENG Yu-xing, LI Dong-sheng
(*Key Laboratory of Science and Technology for Parallel and Distributed Processing, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: A Load Balancing-Supported ID assignment method is the foundation to implement and maintain DHT overlays, realized constant degree DHTs usually use simple pure centralized or distributed ID management strategies, which cannot resolve the contradiction between cost of maintaining topologies' information and topologies' balance. Analyzing the universal tree structures in the topologies, an ID Assignment method RFIDAM based on the internal structure Routing Forest is proposed, which regularly aggregates local balancing information to guide new nodes' joining for overall balance. The experimental results show, with low maintenance and routing message overhead, the system's loading balance is efficiently ensured with the length of IDs differ by at most 2.

Key words: P2P; DHT; ID assignment; constant degree topology; load balance; Routing Foresty

1 引言

由于在实现高效路由的同时具备系统拓扑结构简单、稳定性强、控制信息少等优点[1],常数度 DHT 在近几年逐渐成为 P2P 领域的研究热点,常数度图 de Bruijn 与 Kautz 也成为构建 DHT 理想的基准拓扑候选[2~4].支持负载均衡的 ID 分配方法是实现与维护 DHT overlay 的基础[5],特别对于标准拓扑节点数目不可渐增[2,3]的 de Bruijn 与 Kautz 更为重要.但已有各种基于一致性散列的方法均基于环形或线形拓扑[6~8],其中用于 ID 分配的模型——虚拟二叉树及前驱后继连接在常数度拓扑中并不存在相应的子结构,因此难以在常数度 DHT 中实现.已有常数度 DHT 多使用纯集中式[3]或分布式方法[3,9],前者在系统规模较大时信息维护开销过大、可扩展性差,后者虽不需要维护拓扑信息,但新节点在二次路由时存在盲目性:当没有路径到达时,新节点便不能重定位至理想位置以保证拓扑平衡.因此,在拓扑增长过程中维护越多信息,便可以为新节点分配更合适的 ID,构建更平衡的拓扑,但同时也将负担更大的信息维护开销,即,拓扑信息的维护开销与最终拓扑的平衡程度是 ID 分配方法中的一对矛盾指标.

本文在分析 de Bruijn 与 Kautz 拓扑中通用树结构的基础上,提出常数度 DHT 基于内在结构 Routing Forest 的高效 ID 分配方法,达到了以少量信息开销实现最优的拓扑平衡目标.其思想是利用特定节点聚合拓扑局部均衡信息,通过参考该信息,新加入节点采用一种 ID 二次分配算法选择合适 ID 以控制拓扑中 ID 长度差在 2 以内.与已有方法相比,Routing Forest 有机地结合了常数度拓扑结构特点,达到了良好的平衡效果,可以方便地扩展为 DHT 构建、维护及负载均衡技术.

收稿日期:2009-09-01;修回日期:2010-05-25
基金项目:国家 973 重点基础研究发展计划(No.2005CB321801);国家 863 高技术研究发展计划(No.2009AA01Z142);全国优秀博士学位论文作者专项资金(No.200953);国家自然科学基金(No.60703072, No.60903205);高等学校博士学科点专项科研基金(No.20094307110008)

2 拓扑定义及树结构分析

2.1 De Bruijn 与 Kautz

De bruijn 图 $D(d,n)$与 Kautz 图 $K(d,n)$的顶点集 V 与边集 E 定义如下,其中参数 d,n 分别代表度数与图直径:

$$V_{D(d,n)}=\{x_1x_2\cdots x_n: x_i\in\{0,\cdots,d-1\}\}$$

$$E_{D(d,n)}=\{(x,y): x,y\in V_{D(d,n)}, sub(x,2,n)=sub(y,1,n-1)\}$$

$$V_{K(d,n)}=\{x_1x_2\cdots x_n: x_i\in\{0,\cdots,d\}, x_i\neq x_{i+1}, i\in\{1,\cdots,n-1\}\}$$

$$E_{K(d,n)}=\{(x,y): x,y\in V_{K(d,n)}, sub(x,2,n)=sub(y,1,n-1)\}$$

其中 $sub(s,a,b)$为取字符串 s 从位置 a 到位置 b 子串的操作,如图 1 为 $D(2,3)$与 $K(2,3)$. 定义 $V_{D(d,n)}$ 或 $V_{K(d,n)}$中的完全序关系 $\prec$,对其中任意两节点 x 与 y,有:

$$x\prec y\Leftrightarrow\exists j\in[1,n](\forall i\in[1,j))((x_i=y_i)\wedge(x_j<y_j))$$

$D(d,n)$与 $K(d,n)$均采用移位路由,在此仅介绍长路径路由. 以 x 点到 y 点为例,长路径路由通过 n 次右移实现: $x\rightarrow x_2\cdots x_ny_1\rightarrow\cdots\rightarrow x_ny_1\cdots y_{n-1}\rightarrow y$. 对于 $K(d,n)$,当 $x_n=y_1$ 时,由于节点标识相邻字符必不相同,因此路径中不包括 $x_2x_3\cdots x_ny_1$,只需 $n-1$ 次移位.

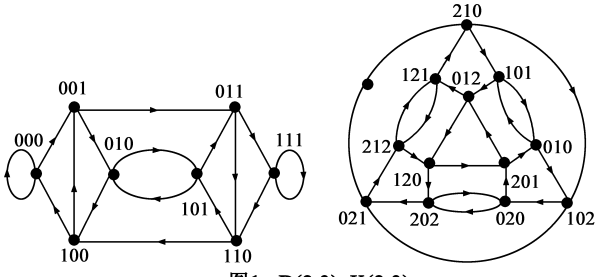

图1 D(2,3), K(2,3)

2.2 ID 生成树与路由树结构分析

2.2.1 ID 生成树

定义 1 ID 生成树规律性地包含了不同直径的标准拓扑 ID. 具体的,第 i 层* 包含直径为 i 标准拓扑中所有 ID,且各 ID 在拓扑增长时由其祖先节点 ID 分裂得到.

$D(d,n)$ID 生成树是标准 d 元树. *level* 0 为一虚拟节点 *root*. 内部节点有 d 个子节点, d 条出边依次标识为 $0\sim d'$,($d'=d-1$). 节点标识由从根到该节点路径中所有边标识的连接组成. $K(d,n)$ID 生成树不同点为:(1) *root* 有 $d+1$ 个子节点 $0\sim d$. 其他内部节点仍有 d 个子节点;(2)树中节点不同于实际 Kautz 节点,但可与以其为序号的 Kautz 节点相互转换. 图 2 为 $D(2,n)$与 $K(2,n)$ID 生成树,后者中括号内外分别为 Kautz 节点与 ID 生成树节点.

ID 生成树的生长形象地反映了拓扑的扩张,ID 分配的目标即为保证 ID 生成树在生长过程中的平衡,减少其叶节点 ID 的长度差,如基于二叉树的 ID 管理[7].

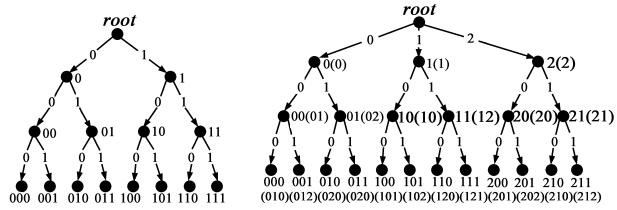

图2 D(2,3),K(2,3)的ID生成树

2.2.2 路由树

定义 2 路由树是拓扑中以任一点为根(*root*)利用移位路由导出的生成树结构. 高度为 n,且 *root* 到任一叶节点的路径是按长路径路由定义所得的两点间的链.

$D(d,n)$路由树是标准 d 元树,所有节点长度为 n. 内部节点有 d 个子节点, d 条出边依次标识为 $0\sim d'$,且子节点标识由父节点标识后 $n-1$ 位连接边标识构成. 因此第 i 层有长度为 $n-i$ 的共同前缀 $sub(root,i+1,n)$,而且包含 $D(d,n)$中所有前缀为 $sub(root,i+1,n)$的节点,第 n 层包含拓扑中所有节点. 每层节点均自左向右按 $\prec$ 序排列.

$K(d,n)$路由树同样为标准 d 元树,与前者区别是:(1)内部节点引出的 d 条边依次标识为 $0\sim d$ 中不同于该节点 ID 末位字符的 d 个值;(2)第 $n-1$ 层包括所有以 *root* ID 末位字符为首的 d^{n-1}个节点,第 n 层包括其余的 d^n 个节点,每层节点按照 $\prec$ 序自左向右排列.

图 3 为 $D(2,3)$中以 110 为根的路由树及 $K(2,3)$中以节点 120 为根的路由树.

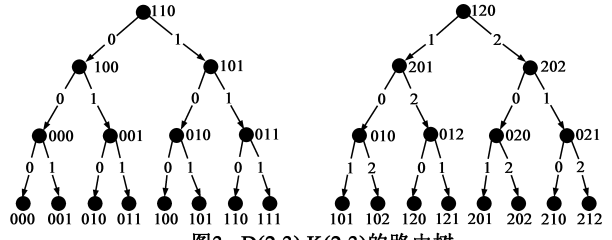

图3 D(2,3),K(2,3)的路由树

可以看出,标准拓扑的路由树和 ID 生成树均按 $\prec$ 序包含节点全集,但直接用于 ID 分配时均有不足:ID 生成树虽能够反映拓扑扩张过程,但其内部节点为虚拟节点;路由树由真实节点构成但结构难以扩展,不适用于包含任意数目节点的非标准拓扑. 因此我们考虑利用路由树中的真实连接关系构建 ID 生成树中所需的

* 在树中,一个节点的层数(*level*)是从根到该节点的路径长度

虚拟连接,实现一种有机结合两种树结构的 ID 分配方法.

3 基于 Routing Forest 的 ID 分配方法

3.1~3.4 节以 de Bruijn 为例介绍 Routing Forest 结构及 ID 分配方法 RFIDAM(Routing Forest based ID Assignment Method),3.5 节介绍针对 Kautz 拓扑的变动.

3.1 Routing Forest 结构

定义 3 Routing Forest(RF)是以节点 0^n 为根的路由树中的一组子树,所有子树的叶节点集不相交且并集为节点全集.子树是 RFIDAM 的最小平衡单元,RFIDAM 通过维护子树平衡达到整棵路由树的均衡.

将子树与其叶节点集合分别记为 T_t 和 $Leaves_t$,其中 t 为该子树的 d 元标识,记其根和高分别 $root_t$ 与 h_t(上下文确定时 t 可以省略).则子树 T_t 包含 d^h 个叶节点.标准 $D(d,n)$拓扑中,RF 包含 d^{n-h}棵类似子树,其标识包括从 0^{n-h}至 d'^{n-h}所有长度为 $n-h$ 的 d 元串.$root_t$ 形式为 $0^h\cdot t$,且 *level* i 节点有共同前缀 $0^{h-i}\cdot t$,*level* h 包含有共同前缀 t 的叶节点集合 $Leaves_t$.如图 4 为 $D(3,7)$RF,其中共有 3^4 棵高为 3 的子树:T_{0000} ~ T_{2222}.每棵子树包含 3^3 个叶节点,如叶节点 2222·000 到 2222·222 将构成 $Leaves_{2222}$.

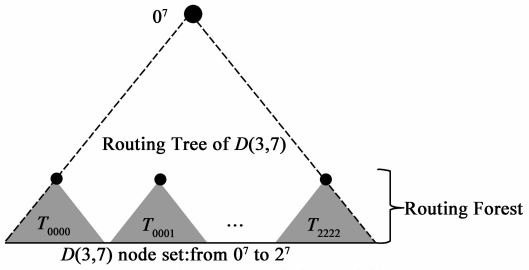

图4 $D(3,7)$中的RF结构

定义 4 平衡因子 δ 为子树或 RF 中叶节点 ID 长度最大值与最小值的差.

ID 分配的目的即为减小 δ(RF).作为最小平衡单元,子树集中式加入节点以维持 $\delta\leqslant 1$,而 δ(RF)与子树 δ 的关系通过子树高度 h 的定义来刻画:h 为该子树最大叶节点长度 n 的单调非递减整数函数,即 $h(n)\in(0,n]$.如 $h(n)=n$,则 $\delta(\mathrm{RF})\leqslant 1$,拓扑达到最佳均衡;如 $h(n)=1$,路由树为完全随机树,拓扑均衡性最差.我们基于定理 1 在纯集中式策略与纯随机无策略间取得最佳折衷:

定理 1 d 元 RF 中,如果 $h(n)=\min(n,k\lceil\log_d n\rceil)$,则可以大概率* 保证 $\delta(\mathrm{RF})\leqslant 2$,其中 k 为一合适常数.

证明:文献[7]中定理 2.1 证明对于二叉树,如 $h(n)>\min(n,\log_2 n+c)$,则大概率 $\delta(\mathrm{RF})\leqslant 2$,其中 c 为一合适常数.因此二元 RF 选择 $h(n)=\log_2 n+c=k'\log_2 n$ 可保证 $\delta(\mathrm{RF})\leqslant 2$,此时子树与 RF 叶节点数量比 $R=2^{h-n}=n2^{k'-n}$.R 的数学意义为:随机向一区间内添加节点,则任两个 R 比例子区间内的节点数量大概率相差不超过 $2^2=4$ 倍.对于高度 n 的 d 元树,R 比例的叶节点 $N=d^nR$ 需要由高度为$\lceil\log_d N\rceil=\lceil n+k'\log_d n-n\log_d 2\rceil<k\log_d n$ 的 d 元子树管理.当 $h(n)=n<k\lceil\log_d n\rceil$时,由前述分析可知此时 $\delta(\mathrm{RF})\leqslant 1$;当 $h(n)=k\lceil\log_d n\rceil<n$ 时,同样可保证 $\delta(\mathrm{RF})\leqslant\log_d 4\leqslant\log_2 4=2$,定理得证.

RFIDAM 由节点信息聚合、ID 二步分配算法及 Forest 结构维护三部分组成,其作用分别为:利用各子树 *root* 与内部节点聚合叶节点的相应信息;参考聚合信息为新加入节点选择合适位置以使子树 $\delta\leqslant 1$;在 n_t 变化时维护 RF 结构,保证子树满足定理 1 要求.

3.2 节点信息聚合

定义 5 叶节点 x 的兄弟节点集 $siblings(x)$包含 x 在 ID 生成树中同父节点的 d' 个兄弟节点,它们 ID 长度均为$|x|$且仅末位字符不同.

定义 6 叶节点 x 为饱和的(saturated)指所有兄弟节点在子树中均有物理节点与之对应或已经分裂.

定义 7 叶节点 x 为可分裂的(fissionable)指 x 非饱和或者其所有邻居 ID 长度都不短于$|x|$.即

$$fissionable(x)\Leftrightarrow\forall\, y\in\Gamma(x)(|x|\leqslant|y|)\vee!\ saturated(x),$$

其中 $\Gamma(x)=\Gamma^+(x)\cup\Gamma^-(x)$为 x 邻居集,$\Gamma^+(x)$、$\Gamma^-(x)$分别指其出边与入边邻居集.

在初始化阶段,节点根据 ID 长度 n 计算 $h(n)$,进而确定所属子树及是否为其他子树内部节点.然后每个节点在向上层发送的心跳信息中添加是否饱和叶节点、是否可分裂叶节点信息,内部节点还包括以自己为祖先的叶节点聚合信息,包括最大叶节点长度 $n_{\max}$,叶节点数目 N_l 以及可分裂叶节点数目 N_{fl},即对于 T 中节点 x:

(1)如为**叶节点**,则报告给子树父节点:$n_t=|x|$、$N_l=1$、N_{fl}(若 *fissionable*(x)为真则为 1,否则为 0).

(2)如为**内部节点**,在收到所有子节点的信息后,聚合相应信息并报告给子树父节点:n_t 为子节点中 n_t 的最大值,N_l,N_{fl}为其所有子节点中 N_l,N_{fl}的和.

(3)如为**根节点**,保存所有 *level* 1 节点的聚合信息:本子树高度 h_t、最大叶节点长度 n_t 与叶节点总数 N_t.并定时向 *level* 1 节点发送更新请求.

---

* 指对于节点数量为 N 的系统,概率不小于 $1-O(N^{-a})$,其中 a 为一正常数

由于分裂方式(见 3.3 节)保证节点分裂时保留出边邻居且相邻节点 ID 长度差≤1,因此对任意节点 $x=x_1x_2\cdots x_m$,总存在唯一的信息报告节点 $0x_1x_2\cdots x_{m'}\in\Gamma^-(x)$,$(m'=m-2,m-1,m)$.如果根节点先于叶节点分裂,为保证根节点唯一性,定义按≺序最小的节点为根节点,其兄弟节点接收聚合信息需重定向至该节点.

3.3 ID 二步分配算法

为高效利用聚合信息,RFIDAM 采用一种 ID 二步分配算法.新加入节点 p 在加入网络时将首先确定其所在子树,然后由子树依据聚合信息分配节点位置:

(0)p 联系 gate 节点 g 并生成随机 ID 串 x

(1)确定所属子树并路由至子树根节点

g 根据$|g|$计算 $h=h(|g|)$,进而确定节点 p 加入的子树标识 $t=x_1x_2\cdots x_{|g|-h}$及 $root_t=0^h\cdot t$.实际网络中,p 节点所属子树 T_t 的高度 h_t 与 h 可能有不大于 1 的偏差,此时当到达 $0^h\cdot t$ 后需要修正.如果 *root* 或子树已分裂,则选择路由至所有有 $0^{ht}\cdot t$ 前缀节点中按≺序的最小节点.

(2)自根路由至合适的叶节点位置并分裂

根据聚合信息,p 自根向下递归的选择可分裂节点 N_{fl}不为 0 且叶节点 N_l 最少的分支.当到达某一 *level h*-1 内部节点后,p 点选择其子节点中出边邻居数最大的可分裂叶节点 q 进行分裂,从而成为子树的新叶节点.

选择可分裂叶节点保证了相邻节点 ID 长度差≤1,同 FissionE[2]、SKY[9]一样可以限制拓扑不均衡程度.根据 q 是否饱和,分裂分为饱和分裂和非饱和分裂:

当 q 点为非饱和叶节点时,q 节点分裂后不改变 ID,而 p 点成为 q 节点的新兄弟节点,此时 n_t 不改变,因此不引起 T 结构变化;当 q 点为饱和叶节点时,q 节点分裂成长度为$|q|+1$的 $q\cdot0$ 与 $q\cdot1$.分别成为 q 点与 p 点的新 ID.此时如果使得 n_t 增 1,则将带来子树 T 的增高或分裂.

图 5 为 $h=2$ 的 3 元子树中三个节点的加入过程.$p1$、$p2$ 的加入属非饱和分裂,其标识 212 与 222 分别为非饱和节点 211 与 221 分裂所得.此时子树达到饱和,因此 $p3$ 加入后选择叶节点 210 进行饱和分裂,分裂后 210 节点 ID 变为 2100,而 $p3$ 标识为 2101.子树 n_t 由 3 变为 4.

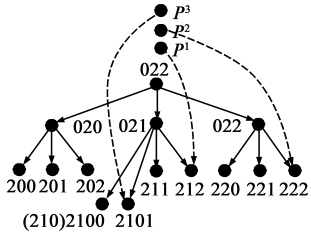

图5 非饱和分裂与饱和分裂

路由连接建立的原则类似于 FissionE[2],即分裂节点将保留原节点所有入边连接,而划分出边连接,这种方式的优点是节点分裂后仍保证聚合路线畅通.

3.4 Forest 结构维护

对于一棵子树 T,定义其 *root* 的两种状态 *thin* 与 *full*,分别表示叶节点未饱和、均已饱和,节点加入子树后将分别产生非饱和分裂、饱和分裂且 n_t 增 1.

记饱和分裂前 $n_t=n$,分裂后 $n_t=n+1$,则原长度为 n 的 d^h 个叶节点共可分裂出长度为 $n+1$ 的叶节点共 d^{h+1}个,此时 T 的变化分两种情况:(1)若 $h(n+1)=h(n)$,T 高度不变故只能容纳 d^h 个叶节点,因此 T 将分裂为 d 棵高度为$h(n)$的新子树,其 ID 分别为 $t\cdot0$ 至 $t\cdot d'$;(2)若 $h(n+1)=h(n)+1$,T 高度增 1,可容纳 d^{h+1}个叶节点,因此 T 将向上延伸增加高度.

root 节点采取如下算法,在接收新节点时判断是否分裂或增加高度以维护 Forest 结构:

```
When a new node join, root judge its status:
Case (thin root):
  If (N_t + 1 = d^h) then root.status = full //子树达到饱和
Case (full-root):
  Case (h(n+1) = h(n)): //子树分裂为 d 棵非饱和子树
    Divide T into d trees with root.status = thin
  Case (h(n+1) = h(n)+1)://子树高度增加 1
    h ++
    select min(x):prefix(x) = 0^h·t to be new thin root
  Notify all nodes in T the adjustment, include new root, h
```

3.5 Kautz 拓扑 ID 分配方法

与 de Bruijn 拓扑相比,Kautz 拓扑的不同在于节点 ID 相邻字符不同,因此 Kautz ID 分配方法中子树的划分、增长与 de Bruijn 存在以下几点不同:

(1)0 前缀节点在上传聚合信息时选择前缀 1 的入边邻居,非 0 前缀节点则选择前缀为 0 的入边邻居.

(2)标识前缀为 0 的子树 T_t 中,*level h* 节点有共同前缀 t,*level h*−1 中节点有共同前缀 $1\cdot t$,*level h*−*i* 节点有共同前缀$\beta_{h-i}\beta_{h-i-1}\cdots01\cdot t$,其中 j 为偶数时 $\beta_{h-j}=0$,否则 =1;标识前缀为 1 的子树正好与前者相反,*level h*-1 中节点有共同前缀 $0\cdot t$,*level h*−*i* 节点有共同前缀$\beta_{h-i}\beta_{h-i-1}\cdots10\cdot t$,其中 j 为偶数时 $\beta_{h-j}=1$,否则 =0.

(3)子树增高时,若原 *root* 前缀为 0,则新根节点为 $1\cdot root$,否则为 $0\cdot root$.

4 实验分析

在 peerSim 模拟器中实现 RFIDAM,实验以不足 1000 节点的标准 de Bruijn 拓扑为初始拓扑,随机加入节点直至拓扑规模达到 10000.

图 6 为不同度数下 k 增长与拓扑最高 δ 的关系,图中曲线为十次实验的平均值.可以发现度数较大时所需 k 值较小,如度数为 7 或 8 时,$k=0$ 或 0.2 时即可达到 $\delta=2$,而当 k 取 1.4 时,对所有度数均已有较好的效果.

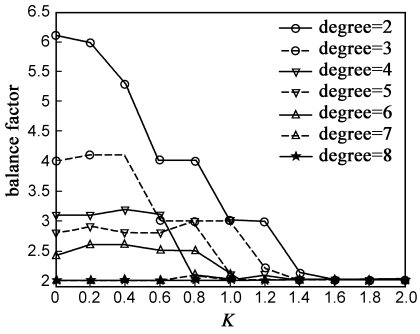

图6　k与δ之间的关系

表 1 为不同度数与 k 下,初始时与最终时的 ID 最大长度 n 与子树的高度 h,表中每一项数值为"初始 n/初始 h/最终 n/最终 h".可以看出聚合信息的子树层数与节点数目并不多,如 $d=2$, $k=1.4$ 时最终子树高度为 6,节点数≤64.而对于高度数,最终子树高度均不超过 3 层.

表 1　不同 d, k 值下的 n 与 h

k	1.4	1.6	1.8	2.0
$d=2$	11/5/14/6	11/6/14/7	11/7/14/7	11/7/14/8
$d=3$	7/3/9/3	7/3/9/4	7/4/9/4	7/4/9/5
$d=4$	6/2/8/3	6/3/8/3	6/3/7/3	6/3/7/3
$d=5$	5/2/7/2	5/2/7/2	5/2/6/3	5/3/6/3
$d=6$	5/2/6/2	5/2/6/2	5/2/6/2	5/2/6/3
$d=7$	4/1/5/2	4/2/5/2	4/2/5/2	4/2/5/2
$d=8$	4/1/5/2	4/2/5/2	4/2/5/2	4/2/5/2

图 7 为 $k=1.4$ 及不同度数下,拓扑规模增长过程中不同长度节点数目占总节点数的比例变化情况,每 200 节点记录一次.可以看出每种拓扑均有多种节点长度出现,但由于同时最多存在三种不同长度,因此每种长度只在一定规模内出现.低度数拓扑的这种变化周期较短,高度数拓扑正好相反,这是由于后者分裂中非饱和分裂比例大,节点长度更容易保持均衡.

5　结论

本文 ID 分配方法中的聚合信息均为简单的数量值,且通过心跳消息传递,而且聚合子树高度随节点规模 log 量级增大,因此该方法消息开销与节点加入开销较低,且有较好的可扩展性.另外,由于节点分裂后仍保留移位连接的特性,因此在数据资源分配时可以采用简单的前缀数据匹配[2],而非复杂度较高的 Kautz 匹配度[9]等.同时 ID 长度的均衡性决定了采用该 ID 分配方法的 DHT 可以高效地估计拓扑直径以及节点数量,以实现相应的全局计算功能.

研究如何优化内部结构或支持节点的异构性,以及如何扩展该结构为 DHT 构建及负载均衡技术或者与现有 DHT 方法相结合是我们下一步研究的目标.

参考文献:

[1] S Ratnasamy, S Shenker, I Stoica. Routing algorithms for DHTs: some open questions[A]. Proc of IPTPS 2002[C]. Cambridge, USA, 2002. 45 – 52.

[2] D Li, X Lu, J Wu. FissionE: a scalable constant degree and low congestion DHT scheme based on kautz graphs[A]. Proc of INFOCOM 2005[C]. Florida, USA, 2005. 1677 – 1688.

[3] D Guo, J Wu, H Chen, X Luo. Moore: an extendable peer-to-peer network based on incomplete kautz digraph with constant degree[A]. Proc of INFOCOM 2007[C]. Anchorage, Alaska, USA, 2007. 821 – 829.

[4] D Loguinov, A Kumar, V Rai, S Ganesh, Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience[A]. Proc of SIGCOMM 2003[C]. Karlsruhe, Germany, 2003. 395 – 406.

[5] 韩华,代亚非,李晓明. Emergint:一种支持多节点并发动态增删的 P2P 路由算法[J].电子学报,2004,32(9):1579

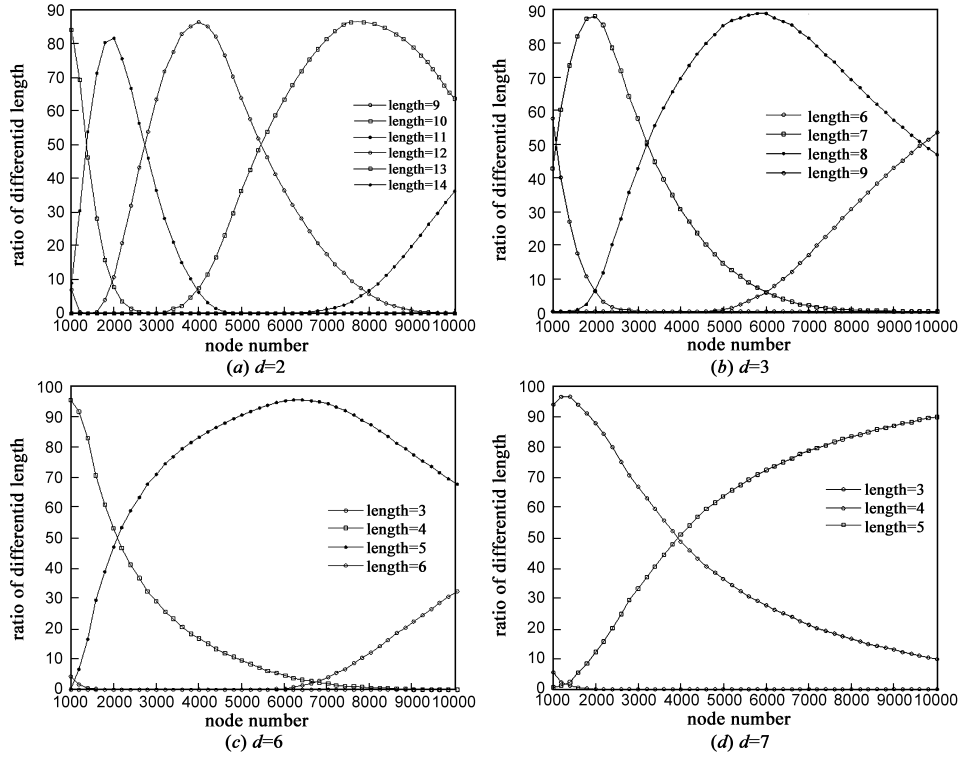

图7　不同节点长度比例随拓扑增长的变化

– 1584.
Han Hua, Dai Ya-fei, Li Xiao-ming. Emergint: a P2P routing algorithm that supports multi-node dynamic concurrent join and leave[J]. Acta Electronica Sinica, 2004, 32(9): 1579 – 1584(in Chinese)
[6] D R Karger, M Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems[A]. Proc of IPTPS 2004[C]. La Jolla, CA, USA, 2004. 131 – 140.
[7] Gurmeet Singh Manku: Balanced binary trees for ID management and load balance in distributed hash tables[A]. Proc of PODC 2004[C]. Newfoundland, Canada, 2004. 197 – 205.
[8] Krishnaram Kenthapadi, Gurmeet Singh Manku: Decentralized algorithms using both local and random probes for P2P load balancing[A]. Proc of SPAA 2005[C]. Las Vegas, NV, USA, 2005. 135 – 144.
[9] 张一鸣. 虚拟计算环境中的高效覆盖网构建技术研究[D]. 长沙, 国防科学技术大学, 2008.
Zhang YM. Research on Efficient Overlay Construction in Virtual Computing Enviroment [D]. Changsha: National University of Defense Technology, 2008. (in Chinese)

作者简介:

王小海 男, 1983 年生于山东省安丘, 博士研究生, 主要研究方向为 P2P 计算、P2P 资源定位技术.
E-mail: seeseahai@163.com

彭宇行 1963 年生, 男, 教授, 博士生导师, 主要研究领域为并行与分布式处理.

李东升 1978 年生, 男, 博士, 副教授, 主要研究领域为计算机网络 P2P 资源定位技术.