第33卷 第20期 计 算 机 工 程 2007年10月
Vol.33 No.20 Computer Engineering October 2007
·软件技术与数据库· 文章编号：1000—3428(2007)20—0067—03 文献标识码：A 中图分类号：TP311

Hash-tree 反碰撞算法

张 虹，韩 磊，马海波
(中国矿业大学计算机学院，徐州 221008)

摘 要：针对 EDFSA 算法标签识别效率低以及二叉树搜索需检测碰撞准确位置等问题，提出了 Hash-tree 反碰撞算法。分析了算法的关键问题，确定了算法策略，进行了算法设计，证明了 Hash-tree 反碰撞算法识别效率期望值在 36.8%~100%之间，优于 EDFSA 算法。仿真验证表明，该算法在识别效率方面有新突破，特别是在识别大量标签时优势明显。

关键词：标签碰撞；反碰撞；无线射频识别技术(RFID)；Hash

Hash-tree Anti-collision Algorithm

ZHANG Hong, HAN Lei, MA Hai-bo
(School of Computer & Technology, China University of Mining and Technology, Xuzhou 221008)

【Abstract】 Against the low efficiency in identifying tags by EDFSA and a demand for testing the precise location of anti-collision by binary tree search, a novel anti-collision algorithm based on Hash-tree is presented. By analyzing key problems of the algorithm and making a strategic plan for it, rate of identifying tags with the algorithm is expected to reach between 36.8% and 1, which outperforms EDFSA. Simulation results show that the algorithm is outstanding in tag identification, especially when dealing with large amounts of tags.

【Key words】 tag-collision; anti-collision; radio frequency identification(RFID); Hash

无线射频识别技术(radio frequency identification, RFID)是一种非接触的自动识别技术，出现于20世纪40年代，发展于80年代。它以RF便携集成电路为载体，通过电磁传播、存储、检索数据。RFID系统一般由电子标签和阅读器组成。阅读器负责发送广播并接收标签发送的标识信息；标签收到广播命令后将自身标识信息发送给阅读器。两个或者两个以上的标签在同一时刻向阅读器发送标识信息时，将产生冲突，解决冲突的方法称为反碰撞算法[1]。

反碰撞算法决定标签识别效率，影响RFID系统性能。EPCglobal组织提出了基于位的二叉树算法和基于ALOHA的算法，ISO组织制定了自适应协议和二叉树搜索算法[2]，ISO的自适应协议和EPCglobal的基于ALOHA算法非常相似。基于位的反碰撞算法需要阅读器检测数据碰撞比特位的准确位置，有一定局限性。诸多学者对基于ALOHA的反碰撞算法做了改进，效果明显的是文献[3]提出的EDFSA反碰撞算法，其标签识别效率期望值可达到34.6%~36.8%之间。随着RFID技术应用领域的不断扩大和用户的急剧增加，碰撞概率大大提高，34.6%~36.8%的识别效率成为RFID系统的瓶颈。为此，本文提出了Hash-tree反碰撞算法，使RFID系统标签识别效率进一步提高。

1 Hash-tree 反碰撞算法

1.1 相关概念

(1)时隙(slot)

标签和阅读器的交互在特定时间片内完成，这个时间片称为时隙。只有一个标签发送信息的时隙称为唯一时隙，同理，多个和没有标签发送信息的时隙分别称为碰撞时隙和空时隙。

(2)帧(frame)

阅读器发送完命令后，等待一组时隙接收标签信息，这组时隙构成一帧，时隙数称为帧长度。文中用三元组$<c_0 \quad c_1 \quad c_k>$表示一帧中空时隙数、唯一时隙数、碰撞时隙数，$L$表示帧长度。

(3)Hash 运算

Hash 运算通常完成从一个数据集合到另一个数据集合的映射。本文通过 Hash 运算完成从标签 ID 集合到帧时隙的映射。

(4)Hash-tree 反碰撞算法

Hash-tree 反碰撞是本文提出的算法。标签通过 Hash 运算选择特定时隙发送自身标识信息，对于碰撞标签，阅读器按树形结构，通过深度搜索碰撞时隙完成对识别域内所有标签的识别，因此，本文称该算法为 Hash-tree 反碰撞算法。

1.2 关键问题解析

反碰撞算法的实质是减少标签碰撞。理想情况是：帧长度和出现在阅读器识别域内的标签数基本相等，并且标签能够均匀地分布在不同时隙发送信息。另外，算法应能够进一步识别发生碰撞的标签。于是，标签时隙分配、帧长度调整以及搜索参数选择是 Hash-tree 反碰撞算法的关键问题。

(1)时隙分配问题

本文算法标签通过 Hash 运算选择帧时隙，而不是随机选择时隙。本文选用式(1)所示的 Hash 函数解决时隙分配问题。

$$\mathrm{Hash}(key)=(\lfloor key/w \rfloor)\%L \tag{1}$$

其中，key 表示标签的关键码；w 为阅读器发送给标签的正整数；为保证分配效果，帧长度 L 一般取质数。例如，若某标签的关键码 key 为 12345，阅读器发送给标签的 w、L 值分别

基金项目：江苏省自然科学基金资助项目(BK2005021)
作者简介：张 虹(1942－)，女，教授、博士生导师，主研方向：自动化，软件工程；韩 磊，硕士研究生；马海波，副教授
收稿日期：2006-10-30 **E-mail：**hongzh@cumt.edu.cn

为 59、19，根据式(1)得 Hash(*key*)等于 0，即关键码为 12345 的标签选择第 0 时隙发送数据。

(2)帧长度调整问题

根据前一帧的识别情况，估计标签数目，动态调整后一帧的帧长度可有效提高标签识别效率。通常有 3 种估计标签数的方法：(1)用公式 $2c_k+c_1+\xi$ 估计(ξ 表示修正值)[4]；(2)用 2.392 $2c_k$估计[5]；(3)用碰撞比例 c_k/L 估计[3]。

Hash-tree反碰撞算法解决帧长度调整问题的策略是：当 $L<2.392\ \ 2c_k \leqq L_{MAX}$时，下一帧的帧长度改变为Prime(2×$L$)，此时，下一帧称为扩展帧，当前帧是下一帧的原始帧；否则，以Little(L)为帧长度开始深度搜索，当前帧称为下一帧的父帧，下一帧称为当前帧的子帧。符号含义见表 1。

表 1 算法符号说明

符号名称	意义
Little(x)	取不大于 x 的质数(无更小的时，取相等)
Generate_queue()	当前帧的碰撞时隙队列
Slot(f)	取第 f 帧的 *Squ* 的队首元素，队列为空时，Slot(f)返回-1
deleteSlot(f)	删除第 f 帧的 *Squ* 的队首元素
Len(f)	第 f 帧的帧长度
Weight(f)	Table 中帧号从 0 到 f 的所有行 w 值乘积
Prime(x)	与 x 最接近的质数(如两个，则取较大者)
SetFlag(f)	将 Table 第 f 帧的 *Flag* 置 1
Flag(f)	返回第 f 帧的 *Flag* 标志
Add_recorder(f, w, L, *squ*, *flag*)	向 Table 中添加记录[f w L *squ flag*]
delete(f)	删除 Table 帧号为 f 的记录
L_{MAX}	系统允许的最大帧长度

(3)搜索参数选择问题

Hash-tree 反碰撞算法通过递归搜索碰撞时隙，识别碰撞标签。递归搜索时，必须适当改变搜索参数，使父帧映射到同一时隙的标签尽可能映射到子帧的不同时隙。

由逆推法知，碰撞标签的关键码的差别是包含不同数目的 $w\times L$，所以，参数 w 的选择原则是：子帧的 w 值是父帧 w 与 L 的乘积。值得说明的是，算法的极限情况为 w 为 2 的 n 次幂(n 取自然数)，L 为 2，相当于比较两个关键码的每一位，不相等的两个数至少有一位不等，所以，递归搜索可以返回。

1.3 Hash-tree 算法设计

(1)通信协议设计

阅读器向标签发送命令字Command(w,L, f, s)，其中，f、s用作标签选择，分别表示帧号、时隙号；w、L用作被选标签Hash运算的参数。标签收到命令后，根据自身变量f_{tag}(帧号)和s_{tag}(时隙号)决定是否或何时响应，原则是：若s等于-1，则f_{tag}等于f或-1 的标签响应；s大于-1 时，只有f_{tag}等于f且s_{tag}等于s的标签响应。阅读器收到标签响应后，发送确认。标签收到确认进入休眠状态，称为已读标签。

(2)数据结构设计

二维表 Table[*Num W FLen Squ Flag*]：阅读器用该表记录帧信息。其中，各列分别表示帧号、w 值、帧长度、碰撞时隙队列、原始帧标志。

阅读器以树形深度搜索标签，当 $c_k=0$ 时，递归返回，当 $L<2.3922c_k\le L_{MAX}$ 时，搜索扩展帧，当 $2.3922c_k\le L$ 时，搜索子帧。递归方程如式(2)：

$$HashTree(f)=\begin{cases} return & c_k=0 \\ HashTree(f+1,extend) & L<2.3922c_k\le L_{MAX} \\ HashTree(f+1,child) & 2.3922c_k\le L \end{cases} \quad (2)$$

(3)算法描述

表 1 给出有关的符号说明。下面从阅读器和标签两部分描述算法，如图 1 和图 2 所示。

```
void HashTree(int L,int w,int f,int s){
    Command (w, L, f, s);
    SQU squ=Generate_queue();//生成碰撞时隙队列
    Add_recorder(f, w, L, squ, 0);
    if (!ck) {
        delete(f);
        if(Flag(--f))delete(f);
        else {
            deleteSlot(f);
            if(Slot(f)==-1) delete(f);
        }
        return;
    }
    else if(2.3922ck>=L&&2.3922ck<=LMAX) {
        SetFlag(f);
        HashTree(prime(2*L),1 ,++f,-1);
    }
    else {
        if (Len(f)>5) L=5; else L=Little(Len(f));
        HashTree(L,Len(f)*Weight(f),++f,Slot(f));
    }
}
```

图 1 阅读器核心算法

STEP1：$f_{tag}=s_{tag}=-1$;
STEP2：接收命令（w, L, f, s）;
STEP3：if (s==-1) { 根据式（1）计算s_{tag} ; 在第s_{tag} 时隙发送数据; }
else { if (f==f_{tag}) {根据式（1）计算s_{tag} ; 在第s_{tag} 时隙发送数据; } }
STEP4：if (收到阅读器确认) {
不再响应同一阅读器的请求; 算法结束;}
else { f_{tag} ++; goto STEP2; }

图 2 标签算法流程

2 算法验证分析

从标签识别效率和标签识别时间两方面分析本文算法与文献[3]中 EDFSA 算法的性能差异；分析标签识别效率的评价模型，证明了本文算法识别效率期望值超越了 36.8%界限；在仿真环境下，验证两种算法在标签识别时间上的不同。

2.1 识别效率分析比较

本节采用文献[3]的算法性能评价标准，定义标签识别效率 a，如式(3)所示。

$$\alpha=\frac{c_1}{L}\times 100\% \quad (3)$$

下面公式中，用n表示出现在阅读器识别域内的实际标签数目；m表示整个标签关键码空间；p_k表示同一时隙被k个标签占有的概率；a_k表示被k个标签占有的时隙数的期望值。在概率论中可以用a_1估计c_1，于是系统效率期望值η 可用式(4)表示。

$$\eta=\mathrm{E}(\alpha)=\frac{a_1}{L}\times 100\% \quad (4)$$

本文运用Hash运算将标签映射到帧时隙内，第 1 帧w为 1，相当于将标签关键码空间分为L组，每组有m/L个标签。k个标签占有同一时隙的情况是由于m/L个标签中有k个标签随机出现在阅读器识别区域内而产生的，因此，第 1 帧p_k表达式为式(5)。由搜索参数改变原则知，以后的各帧关键码空间将不断缩小，其效率问题可从讨论第 1 帧效率函数特性而得到。

$$p_k=\binom{m/L}{k}(L/m)^k(1-L/m)^{m/L-k} \quad (5)$$

由式(4)、式(5)得

$$\eta=a/L=Lp_1/L=p_1=(1-L/m)^{m/L-1}$$

令 $r=L/m$，则

$$\eta = (1-r)^{1/r-1} \tag{6}$$

式(6)求导得

$$\frac{d\eta}{dr} = \frac{(1-r)\left[-r-\ln(1-r)\right]}{r^2(1-r)} e^{(1/r-1)\ln(1-r)} \tag{7}$$

由麦克劳林公式，$-r = \ln e^{-r} \approx \ln(1-r+r^2/2)$，化简式(7)得

$$\frac{d\eta}{dr} = \frac{(1-r)\left[\ln(1-r+r^2/2)-\ln(1-r)\right]}{r^2(1-r)} e^{(1/r-1)\ln(1-r)} > 0 \ (0<r\le 1) \tag{8}$$

$$\lim_{r\to 0}\eta = \lim_{r\to 0}(1-r)^{1/r-1} = e^{-1} = 36.8\% \tag{9}$$

式(8)、式(9)说明，η 单调递增，当 r 趋近于 0 时，η 有最小极限值 36.8%。也就是说，当帧长度远小于标签关键码空间时，识别效率期望值为 36.8%，随着帧长度与标签关键码空间的比例增大，识别效率随之增加。

假设第 2 帧是对碰撞时隙的深度搜索，不是第 1 帧的扩展帧，那么按照帧长度调整策略，第 2 帧的w值应等于第 1 帧的帧长度L，帧长度设为L'，于是p_k，η 表达式如式(10)、式(11)：

$$p_k = \binom{m}{k}\left(\frac{LL'}{m}\right)^k\left(1-\frac{LL'}{m}\right)^{\frac{m}{LL'}-k} \tag{10}$$

$$\eta = L' p_1 / L' = \left(1-\frac{LL'}{m}\right)^{\frac{m}{LL'}-1} \tag{11}$$

显然，$\frac{LL'}{m} > \frac{L}{m}$，前面已经证明$\eta$ 是r 的单调递增函数，所以，第 2 周期的识别效率要高于第 1 周期的识别效率。同理，随着帧号的增大，标签识别效率期望值逐步提高，总体识别效率期望值在 36.8%~1 之间。

EDFSA算法运用帧长度L和标签数n趋于相等时，系统有最大识别效率 36.8%的原理[3]，用分组方式使标签识别效率稳定在 34.6%~36.8%之间。

2.2 识别时间仿真比较

根据 EDFSA 算法和 Hash-tree 反碰撞算法的基本思想，编写了仿真程序。在仿真环境下，EDFSA 初始帧长度为 10，两算法最大帧长度为 256 时，系统识别总时间随着标签数目的变化情况如图 3 所示。当标签数目小于 60 时，EDFSA 算法总体识别时间小于 Hash-tree 反碰撞算法的总体识别时间，这是因为 Hash-tree 反碰撞算法初始帧长度设为 59，标签数小于 59 时，可能浪费时隙。当标签数目大于 60 时，两种算法的系统总体识别时间与标签数目呈线性关系。这说明两种算法，在标签数目大于最大帧长度时，都有分组标签的作用，限制了响应标签数，使识别时间与标签数目保持线性关系。但两条曲线的斜率不同，在标签数目较多时，Hash-tree 反碰撞算法优于 EDFSA 算法，这说明 Hash-tree 反碰撞算法有较高的识别效率。

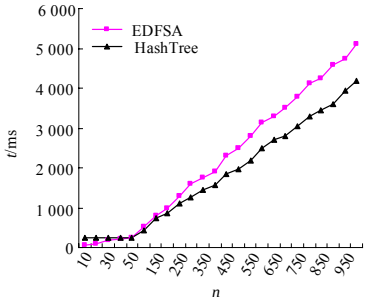

图 3　识别时间t与标签数目n的关系

3　结束语

针对 EDFSA 算法识别效率期望值仅在 36.8%内，造成 RFID 系统的瓶颈问题，以及在采用位二叉树搜索算法时，需要阅读器检测碰撞数据比特位的准确位置等不足，本文提出了 Hash-tree 反碰撞算法。该算法标签识别效率期望值在 36.8%~1 之间，优于识别效率期望值在 34.6%~36.8%的 EDFSA 算法。仿真验证表明，当标签数目大于 60 时，特别是在标签数目很大时，该算法表现出明显优势。另外，本文算法不需要阅读器检测碰撞数据比特位的准确位置，减少了搜索时间，简化了 RFID 系统的软件设计。Hash-tree 算法对解决反碰撞问题有理论指导意义，也势必推动 RFID 系统在矿山跟踪系统、汽车监控、电子支付等领域的应用。

参考文献

1 余松森, 詹宜巨. 跳跃式动态树形反碰撞算法及其分析[J]. 计算机工程, 2005, 31(9).

2 Bawkes P. Anti-collision and Transponder Selection Methods for Grouped “Vicinity” Cards and RFID Tags[Z]. BTG International Ltd., 1999.

3 Lee S R, Joo S D, Lee C W. An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag[C]//Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services. 2005.

4 Vogt H. Multiple Object Identification with Passive RFID Tags[C]//Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. 2002-10.

5 Cha J R, Kim J H. Novel Anti-collision Algorithms for Fast Object Identification in RFID System[C]//Proceedings of the 11th International Conference on Parallel and Distributed Systems. 2005.

(上接第 66 页)

参考文献

1 Tierney B, Aydt R, et al. A Grid Monitoring Architecture[R]. Lawrence Berkeley National Laboratory, Tech. Rep.: GFD-I.7, 2002.

2 Jennifer M, Foster I. Monitoring and Discovery in a Web Services Framework: Functionality and Performance of the Globus Toolkit's MDS4[R]. Argonne National Laboratory, Tech. Rep.: ANL/MCS-P1248-0405, 2005

3 褚　瑞, 肖　侬, 卢锡城. 一种基于GMA 结构的开放式网格资源信息服务[J]. 计算机研究与发展, 2004, 41(12): 2114-2122.

4 Miranda N, Robert S. Grid Computing with Oracle[EB/OL]. (2005-04). http://www.oracle.com/technology/tech/grid/pdf/gridtechwhitepaper_0305.pdf.

5 DataTAG Project Group. Glue Computing Element Schema[EB/OL]. (2003-03). http://www.Cnaf.infn.it/~ser gio/datatag/glue.