Journal of Computer Applications
计算机应用,2012,32(12):3490-3493
ISSN 1001-9081
CODEN JYIIDU
2012-12-01
http://www.joca.cn

文章编号:1001-9081(2012)12-3490-04
doi:10.3724/SP.J.1087.2012.03490

基于优先级分组的防碰撞算法

张从力,彭　璇*,杨　磊
(重庆大学 自动化学院,重庆 400030)
(*通信作者电子邮箱447685478@qq.com)

摘　要:针对在标签数量较多、运动较快的场合,常存在识别效率低且标签漏读率高的问题,提出一种先分组再处理的防碰撞算法——PAJS。该算法按照到达顺序对标签进行分组,以减小漏读率;根据标签识别过程中时隙状况自适应调整帧长度,提高算法的搜索效率;采用跳跃式动态搜索算法处理冲突时隙,从而减少搜索次数和系统传输量。Matlab仿真结果表明,该算法通信复杂度明显小于其他常用算法,吞吐率可达0.59~0.6。在待识别标签较多的场合,该算法优越性更加明显。

关键词:射频识别;标签碰撞;优先级分组;自适应动态帧调整;跳跃式动态搜索
中图分类号:TP391.9　　**文献标志码**:A

Anti-collision algorithm based on priority grouping

ZHANG Cong-li, PENG Xuan*, YANG Lei
(*School of Automation, Chongqing University, Chongqing* 400044, *China*)

Abstract: Concerning the problems of low recognition efficiency and high misreading rate on the occasions of many tags which move fast, an anti-collision algorithm based on packet-first then handling-second was proposed. The algorithm reduced the misreading rate by to grouping the tags according to the order of arriving, it could adaptively adjust frame length based on slot situation to improve the search efficiency, and use jumping dynamic searching algorithm to deal with conflict slots, which could reduce the number of search for readers and the transmission of system. Matlab simulation results show that the algorithm's communication complexity is lower than other commonly used algorithms, and throughput can achieve 0.59~0.6. The larger the number of tags is the more obvious superiority of the algorithm is.

Key words: Radio Frequency IDentification (RFID); tag collision; priority grouping; adaptive dynamic frame adjustment; jumping dynamic searching

0　引言

无线射频识别(Radio Frequency IDentification, RFID)技术中标签通信冲突本质上是多址接入问题,目前常用的两种基于时分多址(Time Division Multiple Access, TDMA)的反碰撞算法有基于ALOHA协议的随机算法和基于二进制树的确定算法。

ALOHA算法[1-4]主要有时隙ALOHA(Slotted ALOHA, SA)算法、帧时隙ALOHA(Frame-Slotted ALOHA, FSA)算法、动态帧时隙ALOHA(Dynamic Frame-Slotted ALOHA, DFSA)算法、自适应动态帧时隙ALOHA(Adaptive Dynamic Frame ALOHA, ADFA)算法等,其中,DFSA算法由于操作简单和性能良好而成为目前最常用的算法之一。树类算法[5-8]大致分为查询树(Query Tree, QT)算法、二进制树搜索(Binary Tree Searching, BTS)算法、动态二进制树搜索(Dynamic Binary Tree Searching, DBTS)算法、回退式索引二进制树搜索(Regressive Index Binary Tree Searching, RIBS)算法等。文献[9]提出的跳跃式动态搜索(Jumping Dynamic Searching, JDS)算法结合了DBTS和RIBS的优点,在减少搜索次数的同时避免了标签ID中多余部分的传输,缩短了数据传输时间。

ALOHA类算法的复杂度和对标签的要求较低,实现简单,但性能不稳定,识别效率不高,可能导致标签饥饿问题。树类算法的识别效率可达100%,但算法相对复杂,需要的存储空间大,识别时间较长,同时,过多的数据交换会导致算法的安全性较差。另一方面,不论是ALOHA类算法还是树类算法,都只有在读写器处理完当前所有被激活标签后,才处理新到达标签,很可能出现后到反而先处理的现象,在标签运动较快的场合,容易造成较高的漏读率。

针对上述问题,本文结合DFSA算法和JDS算法的优点,提出一种基于优先级分组的自适应动态帧时隙跳跃树防碰撞(下文简称PAJS)算法。该算法由两个阶段组成:基于优先级的分组阶段和标签识别阶段。首先,阅读器定时发送分组命令,根据标签的到达顺序对标签进行分组,先到达的标签先处理。然后,阅读器选择一组标签处理,并根据帧时隙算法的思想顺序识别:若当前是空闲时隙,则继续识别下一时隙;若是成功时隙,识别标签;若为碰撞时隙,则利用跳跃式动态树算法识别。在识别过程中,根据空闲时隙、成功时隙、碰撞时隙数的关系和估计的标签数量动态调整帧长度直到整组标签全部处理,继续下一组标签的识别。

收稿日期:2012-06-27;**修回日期**:2012-08-09。
基金项目:中央高校基本科研业务费科研专项研究生科技创新基金资助项目(CDJXS10170011)。
作者简介:张从力(1960-),男,重庆人,教授,主要研究方向:检测与自动化、煤矿安全、智能交通;　彭璇(1989-),女,江西宜春人,硕士研究生,主要研究方向:射频识别、数字语音信号处理与传输;　杨磊(1988-),男(侗族),贵州凯里人,硕士研究生,主要研究方向:矿井无线通信。

1　系统模型

1.1　算法的几点描述

1)采用 Manchester 编码辨认阅读器中数据碰撞的比特位的准确位置 。

2)引入 4 种命令描述算法:

①基于优先级的分组命令 Group(T)。阅读器向标签发送分组命令,收到相同分组数的标签视为一组,并将自身的分组寄存器 g 设为 T。

②选择分组命令 Inquiry(G,L_n)。阅读器发送该命令选择识别标签组,$g = G$ 的标签于 0 ~ L_n 内随机选择时隙。

③时隙选择命令 Tslot(k)。选择了该时隙的标签返回其 ID 信息。

④请求命令 Request($ID_{L_{ID}+1-x}$)。若阅读器发送 $Request$(null,L_{ID}),则查询所有标签;若发生碰撞,则将最高碰撞位 x 置 0,高于该位的值不变,得到查询信息 $ID_{L_{ID}+1-x}$。高位等于 $ID_{L_{ID}+1-x}$ 的标签返回剩余 $x - 1$ 位的值。

⑤选择命令 Select(ID)。选择标签。

⑥读命令 Read(ID)。读取标签信息。

⑦去选择 Unselect(ID)。使标签不再响应该阅读器命令,等待下一次激活命令。

3)用系统吞吐率 S 反映算法的时间复杂度,且假设优先级分组、空时隙、成功时隙以及 PAJS 算法识别过程中阅读器和标签之间信息交换指令所占用的时间都相等;用整个识别过程中读写器发送的总数据量 B_R 和平均单个标签发送的数据量 B_T 来衡量算法的通信复杂度。

1.2　基于优先级的分组

如图 1 所示,圆圈内是阅读器的读写范围。在阅读器内设一加法计数器 T,每隔一定时间向标签群发送一次分组命令,同时计数器 $T + 1$;每个标签设一分组寄存器 g,标签收到的 T 值即为它的分组数 $g = T$,收到同样分组数的标签视为同时到达的一组。阅读器从分组数最小的标签群开始识别,当达到上限值后 T 从 0 开始循环。标签在被正确识别后,记录阅读器地址以便响应其他命令;当标签离开识别范围,标签内分组寄存器清 0,等待下一次命令。

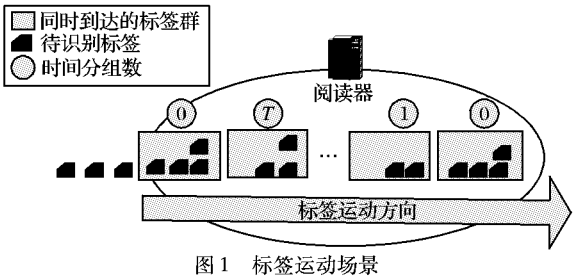


图 1　标签运动场景

1.3　标签识别过程

1.3.1　标签估计和帧长度的调整

动态帧时隙算法大多在一帧结束后进行判决,根据碰撞时隙数和空闲时隙数来调整下一帧长度,当一帧中出现大量冲突时隙则增大帧长度,当一帧中出现过多空闲时隙则减小帧长度,这对不适合的帧反映不及时;而自适应算法是在每个时隙结束后进行判决,这会出现许多无意义的判决,导致帧时隙调整过于频繁。结合两者的思想,PAJS 算法改为每隔若干个时隙判决一次。参照动态帧时隙算法的最大吞吐率一般维持在 0.347 左右,PAJS 算法判决时刻选在 4 的倍数,判决门限设为 0.347。

对未识别标签数量的估计准确度将直接影响到其后的帧长度确定,进而影响整个系统的性能指标。标签估计算法[10-13]中下限值法、Schoute 估计法和空时隙法较简单,但估计准确值不高;Vogt、最大后验概率和贝叶斯估计法较为准确,但计算复杂度偏高。且在这些算法中,冲突时隙中的标签数是未知的,都不能很好地利用这部分信息。

在 PAJS 算法中,冲突时隙是实时处理的,故可记录所有已处理时隙中的标签总数。设一次识别过程中帧时隙总数为 L_{cnt},已识别时隙数为 L_{ide},已识别标签总数为 n_{ide},则剩余标签数:

$$n_{sur} = n_{ide} \cdot (L_{cnt}/L_{ide} - 1) \tag{1}$$

在 2.1 节中证得,当帧时隙长度调整为 $L' = 0.6n_{sur}$ 时,吞吐率最大。

1.3.2　冲突时隙的处理

对动态帧时隙算法的各种改进算法一般都是对标签估计算法的改进,对冲突时隙一般选择不处理。PAJS 算法对发生碰撞的时隙采用 JDS 算法进行处理。阅读器成功识别标签后,就回跳到该节点的父节点继续识别,若发生冲突,则判断碰撞位,生成子节点,要求标签返回有意义的 ID 信息。

JDS 算法的要点如下:1) 阅读器发送 Request(null,L_{ID})命令,要求查询区域内所有标签应答;2) 检测有无碰撞发生,若有则把最高碰撞位置 0,高于该位的数值不变,得到下一次搜索参数(ID_x,N),前 N 位与 ID_x 相同的标签返回信息;3) 若无碰撞则识别单个标签,处理完后删除当前搜索节点,返回父节点,得到下一次搜索参数;4) 重复上述步骤直到识别完所有标签。

JDS 算法在搜索次数和数据交换量方面有很大的改进,但是在标签数量很大的情况下,它需要的数据压栈空间和通信复杂度可能会大大增大。PAJS 算法将它与 DFSA 算法结合处理碰撞时隙,就相当于给标签分了很多组,每次处理的标签数会大大减小,从而减小数据压栈空间和通信复杂度。

1.4　算法步骤

第 1 步　阅读器每隔一定时间发送一次分组命令 Group(T),收到命令的标签分组数 $g = T$。

第 2 步　阅读器向分组值最小的标签群发送查询命令参数 Inquiry(G,L_n)。

第 3 步　分组寄存器 $g = G$ 的标签收到命令后,产生一个不大于 L 的随机数 k,存入时隙寄存器 K。

第 4 步　阅读器发送时隙选择命令 Tslot(k),等待标签返回应答。

第 5 步　若无标签应答,执行下一步;若只有一个标签应答,则成功读取标签信息;若有多个标签返回信息,则利用 JDS 算法处理碰撞标签。监听记录已识别时隙类型及标签数。

第 6 步　判断一组帧是否结束,若未识别完,则执行第 7 步;若一组标签识别完成,$G = G + 1$,执行第 8 步。

第 7 步　当时隙数为 4 的整数倍,判断吞吐率,若吞吐率小于 0.347 则重新调整帧长度,返回第 2 步;否则保持帧长度不变,$k = k + 1$,返回第 4 步。

第 8 步　若 $G = T$,则表明阅读器范围内标签均被识别,等待下一次优先级分组命令;T、G 达到上限值后从 0 开始循环。若 $G < T$,则返回第 2 步。

2 算法分析

2.1 吞吐率分析

定理 1[9]20 阅读器利用 JDS 算法识别 n 个标签所需要的问询次数 $S(n) = 2n - 1$。

证明：

1）当 $n = 1$ 时，显然只需要问询一次，$S(1) = 1$。

2）当 $n = 2$ 时，对于任意两标签，第一次问询确定碰撞位，则两个标签中一个碰撞位为 0，另一个为 1；第二次将最高碰撞位置 0，可识别最高碰撞位为 0 的标签；第三次识别另一个标签。所以 $S(2) = 3$。

3）假设识别 n 个标签需要询问次数 $S(n) = 2n - 1$。当增加一个标签，为了将它和某个匹配度最高的标签区分开来，会增加一次碰撞的识别命令和一次识别该增加标签的命令。故搜索次数 $S(n+1) = S(n) + 2 = 2(n+1) - 1$，结论成立。

假设识别过程中空闲时隙数为 C_0，成功时隙数为 C_1，碰撞时隙数为 C_k。在 PAJS 算法中，假设 C_k 个碰撞时隙对应的标签数为 $n_1, n_2, \cdots, n_{C_k}$，识别这些碰撞标签需要的总时隙数：

$$Nnum_k = \sum_{i=1}^{C_K}(2n_i - 1) = 2\sum_{i=1}^{C_K} n_i - C_k \tag{2}$$

若 C_k 个时隙中包含 n_k 个碰撞标签，则：

$$Nnum_k = 2n_k - C_k \tag{3}$$

又 n_k 等于总的标签数 n 减去 C_1，故：

$$Nnum_k = 2(n - C_1) - C_k \tag{4}$$

若帧长度为 L，标签数为 n。由于标签被分配在每个时隙里的概率均为 $1/L$，可推出在每个时隙内存在 t 个标签的概率为：

$$P(x = t) = C_n^t \left(\frac{1}{L}\right)^t \left(1 - \frac{1}{L}\right)^{n-t} \tag{5}$$

则：

$$C_0 = L \cdot C_n^0 \left(\frac{1}{L}\right)^0 \left(1 - \frac{1}{L}\right)^{n-0} = L\left(1 - \frac{1}{L}\right)^n \tag{6}$$

$$C_1 = L \cdot C_n^1 \left(\frac{1}{L}\right)^1 \left(1 - \frac{1}{L}\right)^{n-1} = n\left(1 - \frac{1}{L}\right)^{n-1} \tag{7}$$

$$C_k = L - C_0 - C_1 \tag{8}$$

利用 PAJS 算法识别标签的吞吐率 S 为：

$$S = \frac{n}{Nnum} = \frac{n}{n_c + n_s + C_0 + C_1 + Nnum_k} \tag{9}$$

其中：n_c 为阅读器基于优先级对标签分组命令；n_s 为阅读器对一组标签发送的查询命令，与帧时隙调整次数有关；$Nnum$ 为识别所有标签所用的时隙数。

假设 PAJS 算法在执行过程中无重新调整时隙这个过程，即在 L 个时隙内识别完所有标签。根据之前推导的公式可知：

$$S = \frac{n}{2 + C_0 + C_1 + 2(n - C_1) - C_k} = \frac{n}{2n + 2C_0 + 2 - L} = \frac{n}{2n + 2L(1 - 1/L)^n + 2 - L} \tag{10}$$

假设算法在 $L = \lambda n$ 时取得最佳值，则：

$$\lim_{n\to\infty} S = \lim_{n\to\infty} \frac{n}{2n + 2\lambda n(1 - 1/\lambda n)^n + 2 - \lambda n} = \frac{1}{2 + 2\lambda e^{-\frac{1}{\lambda}} - \lambda} \tag{11}$$

可求得，当 $\lambda = 0.6$ 时，S 取得最大值 0.6148。

2.2 通信复杂度

2.2.1 阅读器发送的总数据量

算法中阅读器发送的总数据量 B_R 主要包括：1）基于优先级的分组命令所发送的数据量 B_{R1}；2）阅读器查询命令和选择帧时隙命令所发送的数据量 B_{R2}；3）JDS 算法识别碰撞时隙时阅读器发送的数据量 B_{RJDS}。由文献[14]可知，阅读器利用 JDS 算法识别 N 位 ID 的标签平均每次发送查询命令所包含的参数平均长度为 $1 + N/2$。仍假设算法在一帧内识别所有标签，阅读器发送一次命令数据量为 M 位，标签长度为 N 位，有：

$$B_R = (n_0 + n_s + L) \cdot M + Nnum_k \cdot (1 + N)/2 = (2 + L) \cdot M + (2n_k - C_k)(1 + N)/2 = \frac{(1+N)}{2}\left(2n + L\left(1 - \frac{1}{L}\right)^n - n\left(1 - \frac{1}{L}\right)^{n-1} - L\right) + (2 + L) \cdot M \tag{12}$$

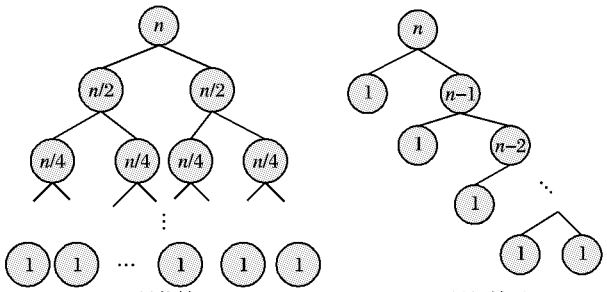
2.2.2 标签发送的总数据量

JDS 算法识别过程中，标签每次发送的数据平均长度也为 $(1 + N)/2$[14]1396。JDS 算法识别标签的过程为一个二叉树结构，所有响应阅读器查询的标签数目可看作二叉树中各个节点值的总和。如果识别 n 个标签构造的是满二叉树，则二叉树的层数为 $\mathrm{lb}\, n + 1$，每层响应的标签数目均为 n，此时标签响应的次数最少，为 $n(\mathrm{lb}\, n + 1)$，如图 2(a)所示。

如果二叉树的所有左子树均为叶子节点，如图 2(b)所示，则构造的二叉树层数为 n，此时响应的次数最大，为 $(n-1) + n(n+1)/2$。故利用 JDS 算法标签响应次数的取值范围为：

$$n(\mathrm{lb}\, n + 1) < R_{JDS}(n) < (n-1) + n(n+1)/2 \tag{13}$$

其中 $R_{JDS}(n)$ 表示利用 JDS 算法识别 n 个标签时，标签响应阅读器的次数。



(a) 最优情况 (b) 最坏情况

图 2 JDS 识别树结构

对于 PAJS 算法，空闲时隙无标签响应，成功时隙标签发送的数据量为 $c_1 N$，碰撞时隙标签发送的数据量为：

$$\frac{N+1}{2}\sum_{i=1}^{c_k} n_i(\mathrm{lb}\, n_i + 1) < B_{TJDS}(n) < \frac{N+1}{2}\sum_{i=1}^{c_k}\left((n_i - 1) + \frac{n_i(n_i+1)}{2}\right) \tag{14}$$

其中 $B_{TJDS}(n)$ 表示 JDS 算法识别碰撞时隙标签发送的数据量。

故 PAJS 算法标签通信复杂度为：

$$B_T = \frac{c_1 N + B_{TJDS}(n)}{n} \tag{15}$$

3 算法仿真

本文选取 DFSA、RIBS、JDS 算法与 PAJS 算法进行比较，用 Matlab 7.4 模拟标签识别过程。阅读器范围内标签数量取 0～250，标签 ID 为 8 位，由函数随机产生。假设标签分布是均匀的，仿真结果在相同条件下，取 20 次实验平均值。

图 3 所示为各个算法的吞吐率比较。DFSA 和 PAJS 算法的初始帧皆取 $L = 8$。PAJS 算法的 λ 取 0.6。

可以看出,DFSA 算法根据估计标签数动态调整帧长度,算法稳定,但吞吐率相对不高。RIBS 算法在每次成功识别后回到父节点继续识别,吞吐率较高。JDS 算法在 RIBS 算法的基础上改进,避免了序列号中多余部分的传输,但吞吐率相较于 RIBS 无变化。PAJS 算法结合 DFSA 算法和 JDS 算法的优点,识别过程中根据估计的标签数量动态调整帧长度,这就增大了成功时隙数;且分时隙使得用 JDS 算法处理的碰撞时隙中标签数量大大降低了,减少了碰撞次数。故本文研究的 PAJS 算法的吞吐率明显高于其他算法,在 $L = 0.6n$ 时可稳定在 0.59 ~ 0.6 左右。

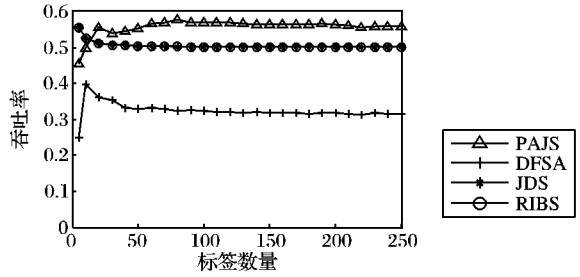


图 3　吞吐率比较

图 4 所示为各算法的阅读器数据通信量比较。设阅读器的查询命令为 8 位。在 DFSA 和 RIBS 算法中,阅读器每次都需要发送完整的 8 位 ID,通信量只与查询次数成比例关系。而在 JDS 和 PAJS 算法中,阅读器只需要发送碰撞及以上 ID 位,大大减小了数据量。

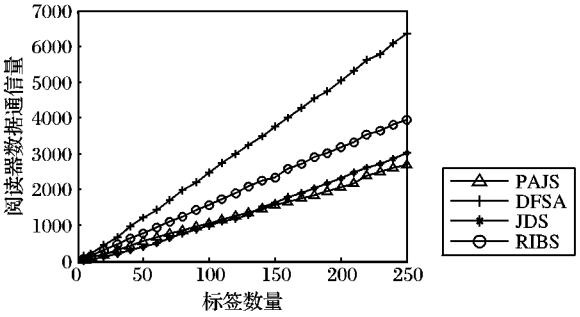


图 4　阅读器数据通信量

图 5 所示为各算法标签数据通信量的比较。碰撞的次数越多,碰撞的标签数量越大,通信数据量就越大。DFSA 算法是通过使帧长度接近标签数量来减少碰撞时隙,从而减少数据通信量;RIBS 算法在多个标签碰撞时,通信量增多;JDS 算法每次只返回有用信息,故通信量少于 RIBS;PAJS 算法自适应动态调整帧长度,降低了标签碰撞概率,同时每个碰撞时隙内标签数量相对于单纯的 JDS 算法也大大减少,标签碰撞数量远小于 JDS 算法。所以 PAJS 算法平均标签数据通信量明显少于其他算法。

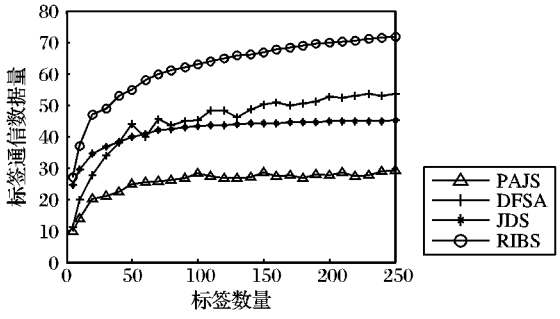


图 5　标签数据通信量

4　结语

在分析传统 RFID 标签防碰撞算法的基础上,结合 DFSA 算法和 JDS 算法的优点,提出一种介于 ALOHA 类算法和树搜索算法间的一种半随机型算法 PAJS。该算法根据到达顺序对标签进行分组,减少了每次处理的标签数量;自适应调整帧长度,可及时处理不合适帧,增大成功时隙的概率;采用 JDS 算法处理碰撞时隙,降低了标签的碰撞概率和数量。仿真结果表明,算法的吞吐率远高于常用的 ALOHA 类算法及树形算法;同时,算法的通信复杂度,尤其是标签数据通信量明显小于其他算法,能很好地适用于标签较多且运动过快的场合。

参考文献:

[1] EOM J B, LEE T J, RIETMAN R, *et al*. An efficient framed-slotted ALOHA algorithm with pilot frame and binary selection for anti-collision of RFID tags[J]. IEEE Communication Letters, 2008, 12(11):861 - 863.

[2] EOM J B, LEE T J. Accurate tag estimation for dynamic framed-slotted ALOHA in RFID systems[J]. IEEE Communication Letters, 2010, 14(1): 60 - 62.

[3] 吴海锋,曾玉. 自适应帧 ALOHA 的 RFID 标签防冲突协议[J]. 计算机研究与发展,2011,48(5):802 - 810.

[4] LIU S, PENG XIAOJUAN. Improved dynamic frame slotted ALOHA algorithm for anti-collision in RFID systems[C]// Proceedings of Advances in Intelligent and Soft Computing. Berlin Heidelberg: Springer-Verlag, 2012: 423 - 430.

[5] 余松森,詹宜巨,彭卫东,等. 基于后退式索引的二进制树形搜索反碰撞算法及其实现[J]. 计算机工程与应用,2004,40(16): 26 - 28.

[6] WANG T P. Enhanced binary search with cut-through operation for anti-collision in RFID systems[J]. IEEE Communication Letters, 2006, 10(4):236 - 238.

[7] LAI Y C, LIN C C. A pair-resolution blocking algorithm on adaptive binary splitting for RFID tag identification[J]. IEEE Communication Letters, 2008, 12(6):432 - 434.

[8] CHEN Z, LIAO M H. An enhanced dynamic binary anti-collision algorithm[C]// Proceedings of the 5th International Conference on Computer Science and Education. Washington, DC: IEEE Computer Society, 2010: 961 - 964 .

[9] 余松森,詹宜巨,王志平,等. 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):19 - 21.

[10] SCHOUTE F C. Dynamic frame length ALOHA[J]. IEEE Transactions on Communications, 1983, 31(4): 565 - 568.

[11] VOGT H. Multiple object identification with passive RFID tags [C]// Proceedings of IEEE International Conference on Systems. Man and Cybernetics. Washington, DC: IEEE Computer Society, 2002:1 - 6.

[12] KHANDELWAL G, YENER A, LEE K, *et al*. ASAP: A MAC protocol for dense and time constrained RFID systems[C] // Proceedings of IEEE International Conference on Communications. Piscataway, NJ: IEEE Press, 2006:4028 - 4033.

[13] HUANG YIHUA, LUI ZONGYUAN, LING GUOJUN. An improved Bayesian-based RFID indoor location algorithm[C]// Proceedings of 2008 International Conference on Computer Science and Software Engineering. Washington, DC: IEEE Computer Society, 2008:511 - 514.

[14] 王亚奇,蒋国平. 基于分组机制的跳跃式动态二进制防碰撞算法[J]. 自动化学报, 2010,36(10):1390 - 1400.