RFID 中基于树的标签防碰撞算法研究

原文链接：https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8128522

摘要

基于树的标签防冲突算法是防冲突算法中的一种重要方法。在本文中，评估了几种典型的树算法。总结了算法的比较，包括时间复杂度、通信复杂度和识别度，指出了每种算法的特点和不足。最后提出了树防碰撞算法的改进策略，并对未来的研究方向进行了展望。

关键词——射频识别；防碰撞算法；电子标签；确定性算法

一、简介

随着物联网技术的飞速发展，射频识别（RFID）技术在生活和生产中的应用越来越广泛。RFID 技术是物联网系统的重要组成部分。它是一种非接触式智能识别技术。与传统的自动识别技术相比，具有体积小、成本低、数据存储量大、安全性高、可重复使用等诸多优点。在工业生产、物流运输和货物管理、商品贸易、健康安全检验和智能交通等领域具有很大的应用价值。RFID 系统主要由阅读器（含天线）、应用系统和大量电子标签[l-3]组成，如图 l 所示。标签主要用于存储被标记对象编码和安全加密的信息，阅读器用于读取、更改和验证标签信息。然而，RFID 系统的应用存在几个问题。

A. 碰撞问题

当多个标签和阅读器在同一信道和信号传输时，由于标签和阅读器之间的相互干扰，会产生碰撞问题。

B. 安全认证和隐私保护

安全问题是当出现窃听攻击和重放攻击时，如何保证标签信息和阅读器的真实性和有效性。

C. 高效存储大量数据

实际应用中的 RFID 设备会不断产生大量数据。如何建立一个高效的数据存储模式降低系统开销，提高查询效率是一个亟待解决的问题。

D. 精确的目标定位和跟踪

如何在室外和室内环境中获取定位对象的准确位置信息将是未来的研究热点。

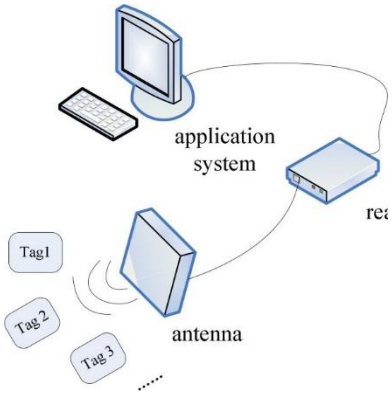

图 1. RFID 系统组成图

碰撞问题是影响 RFID 识别性能的关键问题，制约了大规模标签识别环境下的识别效率。当多个阅读器或多个标签同时以相同频率发送数据时，数据信号会在无线信道中相互干扰，导致数据丢失、丢失、误读等现象。根据碰撞原因的不同，可以分为三类：标签-标签碰撞、

标签-阅读器碰撞和阅读器-阅读器碰撞。对于标签-标签碰撞问题，防碰撞算法分为三类：（1）基于 ALOHA 的非确定性防碰撞算法；（2）基于树结构的确定性防碰撞算法；（3）混合算法。

本文总结了基于树形结构的标签-标签防碰撞算法。从算法的时间复杂度、通信复杂度和识别效率等方面对目前的一些典型算法进行了分析比较。讨论了改进现有算法的途径，并指出了未来的研究方向。这为防碰撞算法树结构的研究提供了参考。

二、几种典型的基于树的防撞算法综述

基于树的防碰撞算法将标签编号转换为由'0'和'I 组成的二进制字符串，当发生碰撞时，根据标签编号分为“0”和“1”两个分支。在每个分支中，算法都会重复查询过程，直到它能够正确识别一个标签。最后，构建了一个树形结构。与随机读取标签的 ALOHA 算法相比，基于树的算法可以避免标签饥饿现象。本节我们选取了目前研究最多的几个基于树的算法，并分析了算法的实现过程和最新的研究进展。

A. BT（二叉树）算法

二叉树算法（BT）是一种基本的二元防碰撞算法，其中每个标签都包含一个计数器和一个随机数生成器[11]。当计数器值为 0 时，标签发送标签信息。当计数器值不为 0 时，标签处于等待状态，不响应，每个计数器的值初始状态为 0。算法步骤如下。

（1）阅读器第一次发出查询命令后，识别范围内的所有标签都响应并发送各自的标签号。

（2）如果发生冲突，每个计数器值为 0 的标签随机生成 0 或 I 并分配给计数器。如果计数器值大于 0，则计数器值增加 I。

（3）如果没有发生冲突，则表明标签已经被识别或产生了空闲节点，所有标签的计数器值减 1。

（4）循环步骤 2，直到识别出所有标签。

图 2 显示了使用 BT 算法识别标签的示例。随着标签数量的增加，查询的数量急剧增加。这导致了大量的空闲时隙，降低了系统识别的效率。否则，BT 算法需要在每个识别周期更新标签计数器的值，并要求标签具有存储和计数功能[12]。结果，它增加了系统的开销。文献[13]在 BT 算法的基础上提出了 ABS 算法。算法中的每个标签都有两个计数器：阅读器的分配槽计数器 Rc 和标签的进度槽计数器 Pc[14]。这两个计数器和随机数使标签具有相同的可识别性，但算法设计得更复杂。识别过程中仍有空闲槽。计数器的值也每次都存储，增加了算法的时间复杂度。江等人[15]引入了一种回退策略和搜索树算法来改进 ABS 算法。该算法消除了空闲时间，减少了识别延迟，但并没有减少系统开销，因为每次查询仍然需要改变计数器的值。

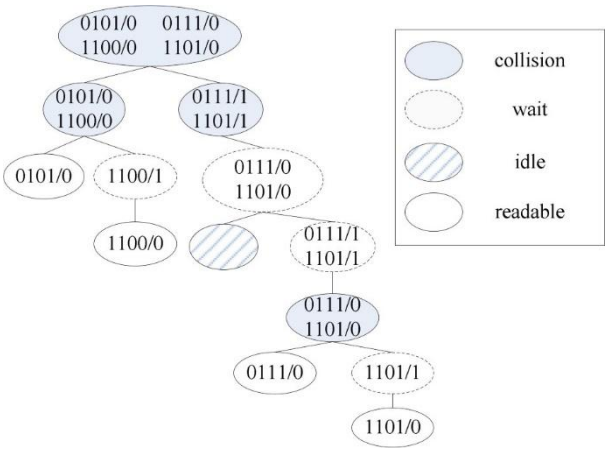

图 2 BT 算法示意图

B. **BS（二分查找）算法**

二叉搜索树防冲突算法（BS）是一种无记忆经典的基于树的防冲突算法[16]。在该算法中，标签不需要存储任何信息，当碰撞发生时，阅读器可以改变查询字符串来识别标签。算法步骤如下。

（1）阅读器以二进制字符串形式向标签发送查询命令。查询字符串长度等于标签号长度，初始值由所有'l'组成。所有要识别的标签在识别范围内响应并将编号发送给阅读器。

（2）如果没有发生冲突，则只识别出一个标签。如果发生碰撞，读取器将查询字符串的最高碰撞位置设置为'0'，并且高于最高碰撞位的编码不变，将最高碰撞到最低的设置为“1”，形成一个新的查询字符串。

（3）阅读器广播一个新的查询字符串，循环第 2 步。当一个标签被识别时，阅读器读取它的编号并将命令发送到“休眠”状态。

（4）从步骤 1 重新开始，直到识别出所有标签。

二进制搜索算法（BS）消除了空闲时隙，不需要存储额外的数据。但是，查询字符串后的每次识别都会从初始状态开始，并且增加了查询次数。同时，当阅读器发送一个二进制字符串时，标签需要发送一个完整的二进制字符串。实际上，阅读器发送的二进制字符串的相同部分不需要发送给阅读器。给定 A、B、C 三个标签{11010010101011011010101101}，使用 BS 算法识别这三个标签的过程如表 l 所示。

表 1.二进制搜索算法识别表

	First search	Second search	Third search
Reqest	11111111	10111111	10101111
Label A	11010010	——	——
Label B	10110110	10110110	——
Label C	10101101	10101101	10101101

基于 BS 算法的改进算法包括动态二分搜索算法(DBS)、动态退避二分搜索算法[20]等。与 BS 算法相比，DBS 算法的查询次数相同，但标签和阅读器之间的数据减少了一半，提高了查询效率。动态二进制退避算法是在 DBS 算法的基础上改进的，在保证 DBS 传输更少数据的同时减少了查询次数。在 BS 算法的基础上，吉林大学的狄春雨[21]提出了 BS-BLG 算法，其主要改进是增加了锁定位组和组分页指令。锁定位组的作用是提取产生冲突的位，并根据连续“1”的数量对组进行优先级排序。它通过分组分页来减少查询次数，并使用后向策略来减少查询次数。BLBO 算法是在 BS 和 DBS 算法[22]的基础上提出的。该算法通过锁定策略锁定碰撞位，然后阅读器只发送碰撞位信息，降低了通信复杂度。采用后向策略，每一个标签被识别后，返回到发生碰撞的节点，BLBO 算法的识别效率接近 50%。文献[23]在 BS 算法和 BLBO 算法的基础上提出了 NBLBO 算法。该算法引入了自适应分叉的思想，通过计算每次碰撞后的碰撞因子来调整下一个查询前缀。相同数量的标签，

C. **QT（查询树）算法**

QT 算法也是一种内存少的算法，每次只需要将标签与阅读器发送的查询前缀进行比较，

将标签与前缀一起发送回阅读器。首先将算法引入栈中，保存查询前缀，提高查询效率。QT 算法步骤如下。

（1）在查询的初始阶段，一个空字符串被压入堆栈，阅读器发送一个空字符串。此时，识别范围内的所有标签都会响应。

（2）如果发生冲突，将最后一个前缀后面的'0'和'1'相加，形成一个新的查询前缀，并将这两个新的前缀压入堆栈。如果没有冲突，则标识一个唯一标签。

（3）如果栈不为空，则弹出栈的查询前缀。收到查询命令后，匹配查询前缀的标签返回余数的代码。如果没有标签响应，就会出现一个空循环，并且会弹出一个查询前缀。

（4）重复以上 2、3 步，直到栈为空，所有标签都被识别。

QT 算法简单易行。它对标签没有额外的存储要求，硬件成本低。但 QT 算法受标签分布影响较大，不预先判断标签碰撞的位置，查询前缀的更新比较机械。在识别过程中会产生大量的空闲时隙。下表列出了标签长度为 12 位且数量分布为随机分布情况下的空闲时隙数量。标签数量从 100 个增加到 600 个。

表 2.QT 算法的时隙表

Number of tags	Query time slot	Idle time slot
100	289	45
200	543	71
300	785	93
400	1031	116
500	1283	142
600	1512	157

可以看出，随着标签数量的增加，空闲时隙也随之增加。查询效率降低，增加了阅读器的能耗。但是 QT 算法还是不可避免的会产生大量的空闲时隙和碰撞时隙。文献[27]在 QT 算法的基础上提出了 MBQT 算法。标签由二进制编码变为多重编码，减少了通信过程中的传输数据量，提高了识别效率。但该算法需要使用阅读器更多的存储空间，而且算法的性能还受标签长度的影响。在[28]中，提出了一种混合查询树算法。该算法根据每个识别周期中的碰撞信息改变查询前缀。与 QT 算法相比，该算法减少了查询次数，将识别效率提高到 59%左右。但是该算法仍然没有解决空闲时隙的问题。周青在 QT 算法[29]的基础上提出了 IHQT 算法，在每个标签中设置一个全加器。该算法使用查询前缀后的三个连续“1”的数量来确定响应阅读器的顺序。这种方法避免了空闲时隙的产生，降低了通信复杂度。在查询前缀之后确定响应读者的顺序。这种方法避免了空闲时隙的产生，降低了通信复杂度。在查询前缀之后确定响应读者的顺序。这种方法避免了空闲时隙的产生，降低了通信复杂度。

D. **CT 算法**

CT 算法是 QT 算法的一种改进算法，由贾晓林博士于 2012 年提出[12]。与 QT 算法不同的是，CT 算法只针对碰撞位置更新查询前缀，减少了不必要的查询，消除了空闲时隙。在 CT 算法中，首先，初始阅读器向标签发送查询指令。如果没有标签冲突，则直接识别，或者阅读器根据第一个冲突位更新查询前缀，即第一个冲突位之前的所有位不变，后面加上“0”和“1”.因此，两个新的查询前缀被压入堆栈。阅读器在查询堆栈顶部发送查询前缀，标签将自己的编号与接收到的查询前缀进行比较。如果一样，CT 算法完全消除了查询过程中的空闲周期，不需要内存标签识别过程中的信息。识别效率大于 50%。在 CT 算法的基础上，提出了一种多周期识别算法（MCT）[30]，将 RFID 多标签识别中读写器与标签通信的识别周期

划分为三个子周期（Q、RO、RI).在算法中，标签根据查询命令确定标志位，根据标志位的值选择响应周期（RO 或 RI）来响应阅读器的查询请求。但是这样一来，阅读器每次发送请求，标签都要判断标志位，根据标志位来划分响应周期，从而大大降低了标签与阅读器的通信效率。在论文[3-1]中，利用双前缀和碰撞位连续性对 CT 算法进行了改进，提出了 DPPS 算法。与 CT 算法相比，平均识别效率提升 36%，平均通信复杂度降低 35.6%。

综上所述，虽然近年来防碰撞算法的研究取得了一些成果，但仍存在算法设计复杂、无法适应实际应用等问题。算法识别效率受标签分布大等影响。一种高效稳定的防碰撞算法是未来研究的重点。

三、几种算法的比较

目前对基于树的防碰撞算法改进的参考文献大多集中在 BS 算法、BLBO 算法、QT 算法和 CT 算法上。在本节中，我们将这些算法的性能与实验分析进行比较

A. 绩效分析

表 3 显示了四种算法的性能参数。其中，n 是要识别的标签个数，k 是标签的个数长度，x 是标签碰撞的位数，lpre 是查询前缀的长度由阅读器发送，nres 是在每个查询周期数中响应阅读器的标签。

表 3 算法性能比较

Anti-collision algorithm	Time complexity	Communication complexity	Recognition efficiency
BS algorithm [32,33]	$n+\log_2(n!)$	$2k*(n+\log_2(n!))$	$1/(1+\log_2(n!)/n)$
QT algorithm [33,34]	$n*(k+2-\lg n)$ (The worst situation)	$(n+1)(2.21k*\log_2 n+4.19k)$	$1/(k+2-\lg n)$ (The worst situation)
BLBO algorithm [22]	$2n-1$	$3k-2x-21+(2x+44)*n$	$1/(2-1/n)$
CT algorithm[35,36]	$2n-1$	$(2n-1)*[l_{pre}+(k-l_{pre})*n_{res}]$	$1/(2-1/n)$

从上表可以看出，BS 算法的时间复杂度为 n+log2(n!)。因为 n！随着标签数量和标签编号长度的增加，时间复杂度和通信复杂度迅速增加，识别效率会越来越低。显然，BS 算法还有很大的改进空间。上表所列 QT 算法的时间复杂度只是最复杂的情况，但可以看出 QT 算法受标签号长度影响很大。当标签号较长时，阅读器的查询范围大大增加，从而增加了空闲时隙的数量。通常，QT 算法的识别效率约为 34%。BLBO 算法和 CT 算法具有相同的时间复杂度，这决定了两种算法的识别效率是一样的。

B. 模拟实验

查询的数量和传输的比特数是反映时间复杂度和通信复杂度的重要指标。为了更直观地比较这些算法的性能，使用 MATLAB 仿真工具对几种算法的查询次数、通信传输位数和识别效率进行仿真。其中，标签个数从 100 个增加到 600 个，增量为 100 个。标签个数是随机分布的，个数为 12 位。仿真结果如图 3 至图 5 所示。

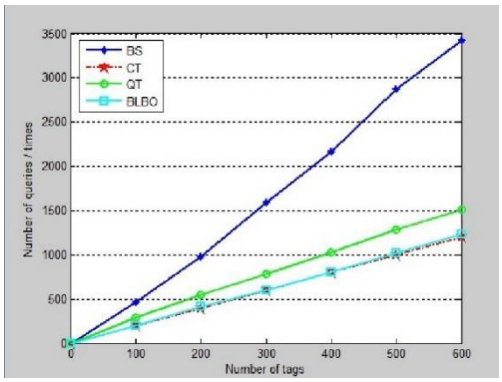

图 3 查询时间比较

从图 3 可以看出，BS 算法在标签数量增加的情况下，阅读器需要发送的查询最多，可以看出 BS 算法最适合在大规模标签环境中使用，并且 QT 算法的查询次数约为 BS 算法的一半，BLBO 算法的性能与 CT 算法相近，优于 QT 算法。

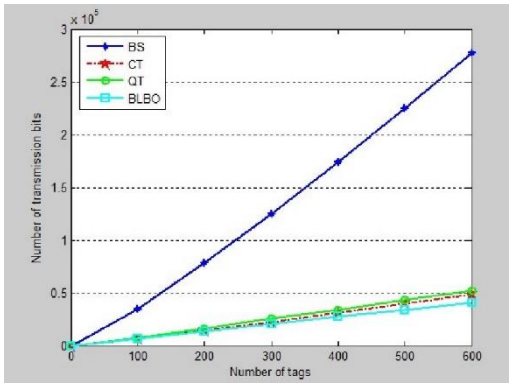

图 4 传输位数比较

在图 4 中，我们可以看到 BLBO 算法和 CT 算法的识别效率在 50%左右，随着标签数量的增加，算法的性能趋于稳定。QT 算法的识别效率在 35%到 40%之间。BS 算法的识别效率最低，随着标签数量的增加，识别效率逐渐降低。

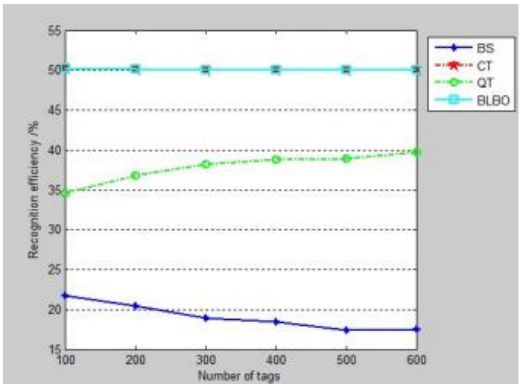

图 5 识别效率对比

当使用四种算法识别标签时，阅读器和标签之间传输的比特数如图 5 所示。从图中可以看出，BS 算法 BS 算法表现最差，而且随着标签数量的增加，需要的数据量增长得越来越快，增加了系统的成本。QT、CT 和 BLBO 算法的传输比特数明显低于 BS 算法。其中，由于在 BLBO 算法中引入了锁回策略，所以表现最好，其次是 CT 和 QT 算法。

四、基于树的防撞算法改进策略

近年来，RFID 多标签防碰撞算法的研究越来越受到众多学者的关注，针对树的防碰撞算法提出了许多改进策略，并取得了一些研究成果。本文在查阅相关文献的基础上，总结出几种比较典型的改进策略。

A. 位锁定

在基于树的防碰撞算法中，如果标签发生碰撞，那么只有碰撞码是未知的。这样，在发生碰撞时锁定碰撞位，然后阅读器只发送碰撞位信息，标签只会返回碰撞位信息，从而消除了冗余数据。如图 6 所示，碰撞后，阅读器的解码信息是 1x0xx000，下一次阅读器只请求 2、4、5 位发送编码信息，所以无论是标签侧还是阅读器都会减少信息存储来提高沟通效率。

在[37]中，DBS 算法采用位锁定的方法进行了改进，使得新算法在传输时延和标签能耗上有很大的改善。在文献[38]中，提出了一种多碰撞联合锁位动态调整算法。该算法使用最高碰撞位和最低碰撞位形成锁位分页指令。最好先估计碰撞槽中的标签个数，然后检测碰撞位的个数，动态选择处理碰撞的算法。新算法的指令成本比 DBS 算法和 QT 算法低 12%和 39%。

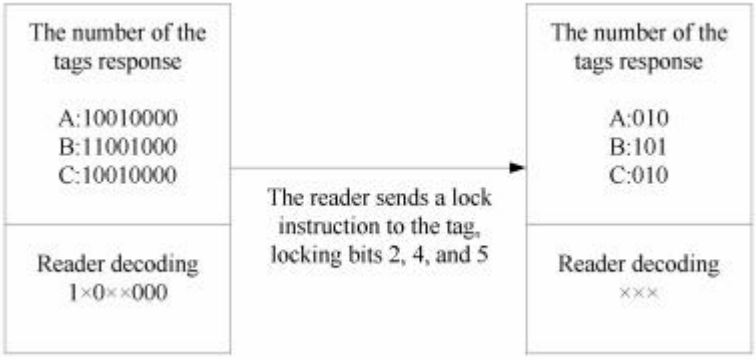

图 6. 锁定位策略图

B. 自适应分岔

现有的几种基于树的防碰撞算法的分叉只是二叉的，导致树的深度过深，增加了查询次数。如果将标签集分配在多个子树上，则可以有效降低查询树深度和标签碰撞概率[22]。在标签识别过程中，经常使用四叉树、八叉树等树分叉算法来减少冲突时隙，相应地，空闲时隙的数量增加。一些研究人员提出了自适应分岔算法来减少空闲时隙。丁志国先生提出了一种自适应防碰撞算法。算法中引入了碰撞因子，碰撞因子是碰撞槽中的碰撞位与标签响应位的比值：

$$\mu = \frac{n_c}{n}$$

当标签发生碰撞时，计算碰撞因子。当 u<0.75 时选择二叉树根据第一个冲突位生成两个查询前缀；u>=0.75 时使用四叉树根据前两个冲突位生成四个查询前缀。为了避免使用四叉树生成空闲时隙，文献[40]提出了一种由阅读器发送检测命令确定具体值来优化四元查询前缀的方法，而任[41]提出了一种利用异或运算消除之前空闲时隙的方法。四元查询前缀的确定. Wang[42]首先使用 MLE 算法估计标签个数，然后使用提出的 CBGN 算法推导出最优分组系数在不同的树下。新算法消除了空闲时隙，显着降低了通信复杂度。

C. 反向传播

在传统的 BS 算法中，标签识别成功后，需要返回根节点获取查询命令。在后向策略中

识别出一个标签后，会回退到该标签节点的父节点或其他非根节点，这样可以大大减少搜索次数，同时也可以减少空闲时隙和冲突时隙的数量。最典型的算法是潘等人提出的智能遍历算法——STT[43]。该算法在碰撞时隙中采用前序遍历节点，空闲时隙采用退到上层节点的机制，有效减少了碰撞次数和空闲时隙。在[44]中，提出了一种增强的 STT 算法，可以根据上一次查询自适应调整查询字符串的长度。与 STT 算法相比，查询总数减少了约 6%。在论文[45]中，在阅读器的查询指令中加入了反向分页，将标签信息分为四部分，减少了不必要的反向查询路径，减少了通信量。

五、结论

基于树的防碰撞算法具有稳定性好、识别效率高等优点，但也存在读写器与标签之间通信数据量大、系统时延高、对标签数量分布影响大等缺点。但基于 Aloha 的算法适应性强、系统开销低，因此基于树算法和 Aloha 算法的混合防碰撞算法将是未来的研究热点。此外，针对移动状态和捕捉效果的标签识别也是未来的研究方向。