基于查询树的防碰撞算法性能分析与研究

李 川[1,5],苏 健[2,4],刘克雄[3],韩 雨[4],赵红军[5]
(1. 绵阳职业技术学院信息工程系,四川绵阳 621000;2. 南京信息工程大学计算机与软件学院,江苏南京 210044;
3. 北京林业大学工学院,北京 100083;4. 电子科技大学通信与信息工程学院,四川成都 611731;
5. 西南科技大学信息工程学院,四川绵阳 621010)

摘 要: 多标签碰撞问题严重影响射频识别系统的性能. 基于查询树的防碰撞算法作为一种确定性算法被广泛的应用于各类射频识别场景中. 本文分析了主流查询树防碰撞算法的性能与不足,并基于查询树方法提出了一种时间有效的防碰撞算法. 该算法基于传统查询树识别模型,实施了一种双查询前缀匹配方法,可以消除传统查询树方法中的空闲时隙. 此外,提出的算法可以充分利用碰撞时隙来提高识别效率. 理论分析和仿真结果表明该算法优于现有的查询树防碰撞算法.

关键词: 射频识别; 防碰撞; 查询树方法; 时间效率
中图分类号: TP391.4 **文献标识码:** A **文章编号:** 0372-2112 (2018)11-2671-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.11.014

Performance Analysis and Research of Anti-collision Algorithms Based on Query Tree

LI Chuan[1,5], SU Jian[2,4], LIU Ke-xiong[3], HAN Yu[4], ZHAO Hong-jun[5]
(1. *Department of Information Engineering, Mianyang Polytechnic, Mianyang, Sichuan* 621000, *China*;
2. *School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, Jiangsu* 210044, *China*;
3. *School of Engineering, Beijing Forestry University, Beijing* 100083, *China*;
4. *School of Communication and Information Science, University of Electronic Science and Technology of China, Chengdu, Sichuan* 611731, *China*;
5. *School of Information Engineering, Southwest University of Science and Technology, Mianyang, Sichuan* 621010, *China*)

Abstract: Multiple tags collision problem severely impairs identification performance of RFID systems. Anti-collision algorithms based on query tree have been widely applied in various RFID scenarios. In this paper, we analyze the performance and deficiency of mainstream query tree based algorithms. And then, a time efficient anti-collision algorithm based on query tree is proposed to enhance the identification efficiency. Specifically, the dual prefixes matching method is implemented based on the conventional query tree identification model, which can significantly remove idle slots. Moreover, the proposed method can also make extensive use of collision slots to improve the identification efficiency. Both theoretical and simulation results indicate that the proposed algorithm outperforms the existing query tree based algorithms.

Key words: radio frequency identification (RFID); anti-collision; query tree (QT); time efficiency

1 引言

射频识别(Radio frequency identification, RFID)是一种用于目标自动识别的无线通信技术. 一个典型的RFID 系统由一个读写器和多个低成本和体积小的标签组成,每个标签对应唯一的标识符(unique identifier, UID)并贴在某个物体上,读写器通过双向通信识别标签从而实现目标物体的自动识别[1]. 当多个标签同时向读写器返回 ID 数据时会产生“多标签碰撞”问题,这会导致读写器无法成功识别标签. 为了解决这一问题,读写器必须运行相应的防碰撞算法来识别多标签,特别是标签数量密集的 RFID 环境.

现有的 RFID 多标签防碰撞算法主要可以分为三

收稿日期:2017-09-19;修回日期:2018-04-26;责任编辑:蓝红杰
基金项目:四川省科技计划面上项目(No. 2018JY0337);江苏省高等学校自然科学研究面上项目(No. 17KJB510036)

类,即:概率性算法[2,3],确定性算法[4~6]和混合型算法[7,8].概率性算法基于 Aloha 思想,应用最普遍的是动态帧时隙 Aloha(Dynamic framed slotted Aloha,DFSA)算法.DFSA 算法的原理是,读写器在每轮识别周期指定帧长 F(帧长度大小等于时隙个数),然后向其工作域内的标签广播查询命令,接收到查询命令的标签在 1-F 间随机选择一个时隙来响应读写器并返回自身的 ID 数据.读写器检测标签响应并统计不同状态下的时隙个数.当一帧结束后,读写器根据此轮识别周期的统计结果来估计剩余标签数并调整相应的帧长,直到所有的标签被成功识别.因为遭受‘标签饥饿’问题,概率性算法并不能确保 100% 的识别率.确定性算法又叫做查询树(query tree,QT)算法,对于这类算法而言,比特追踪技术[6,7]能够使读写器识别 ID 数据碰撞的具体位置,因此被广泛的应用于 QT 防碰撞算法,诸如碰撞树(collision tree,CT)[4]算法,连续碰撞比特映射算法(consecutive collision bit mapping algorithm,CCMA)[5],Q 进制搜索方法(q-ary search scheme,QAS)算法[9],查询窗树形(Query window tree,QwT)[10,11]算法以及它的进化算法.与传统的 QT 算法相比,上述算法可以实现更好的性能.然而,它们的性能高度取决于标签的 ID 分布和碰撞比特位置.混合防碰撞算法融合概率性算法和 QT 算法来实现更好的性能,代价是复杂的读写器和标签设计[12].QT 防碰撞算法的特点是设计简单,无需估计标签数,可以保证所有的标签都被成功识别.

本文主要讨论 QT 防碰撞算法,阐述了已有工作的原理和性能,分析了目前存在的问题和不足,提出了一种双前缀探测方法(dual-prefix probe scheme,DPPS)来提高 RFID 系统的识别效率.基于 DPPS,多个标签可以在同一时隙内被成功识别.图 1 所示为传统 QT 算法和 DPPS 算法的系统传输模型.本质上,读写器可以利用单次查询命令在同一个时隙探测两个标签.因此,DPPS 算法能够节省总的识别时间.除此之外,不同于已有的多比特碰撞仲裁机制[5,10],DPPS 能够充分利用碰撞时隙来识别标签并彻底消除了多进制查询产生的空闲时隙.理论分析和仿真结果表明,我们提出的 DPPS 算法在识别效率、通信复杂和系统效率上均优于主流 QT 算法.

2 相关工作

2.1 比特追踪技术

比特追踪技术通常基于曼彻斯特编码[4,5,10~13],它将单比特数据编码为电平跳变.数据 0 被编码为‘正-负跳变’,数据 1 被编码为‘负-正跳变’,反之亦然.由于曼彻斯特编码中不存在‘无跳变’状态,因此读写器可以识别并追踪某个碰撞比特.通过这种编码方式,读写

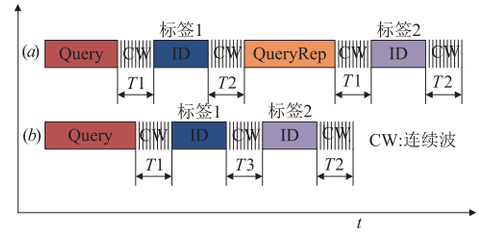


图1 系统传输模型:(a) QT算法 (b) DPPS算法

器可以在接收端解码并检测出接收信号的比特信息.图 2 所示为曼彻斯特编码的一个示例.两个标签的 ID 分为别‘0101’和‘0011’.当 2 个标签采用曼彻斯特编码方式同时向读写器发送它们的 ID 数据时,读写器会接收到数据‘0xx1’,其中‘x’表示碰撞比特.从这个例子中可以看出碰撞比特的位置为第 2 和第 3 位.读写器检测到的碰撞信息有助于将碰撞的标签分组并更快的识别它们.

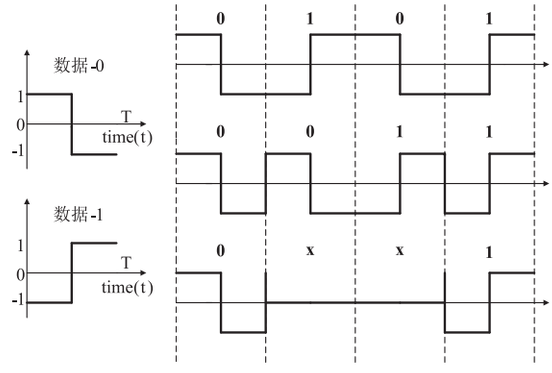


图2 曼彻斯特编码示例

2.2 CT 算法

CT 算法的基本原理是[4]:读写器检测接收到的数据的最高碰撞位,并将其分别置为 0 和 1,产生新的查询前缀并在下一个时隙发送新的查询命令,接收到查询命令的标签只需要返回与前缀匹配后的剩余 ID 数据.同传统 QT 算法相比,CT 算法消除了查询过程中的空闲时隙,并减少了时隙内的数据传输量.假设当前读写器工作域内存在 5 个标签(A,B,C,D,E),其 ID 分别为‘00001011’,‘00110110’,‘10011000’,‘11001101’和‘01100100’,采用 QT 算法识别,其识别过程如表 1 所示.

从表 1 中可以看出 CT 算法识别 5 个标签共需要 9 个时隙.QT 类防碰撞算法的识别过程类似一颗二进制遍历树,每个标签都可以通过对树的遍历而被探测[5],树中的根节点到叶子节点的路径对应标签 ID 的每个比特,且每个节点具有空闲,成功和碰撞三种状态.由于 CT 算法在 QT 算法的基础上消除了空闲节点,所以 CT 算法所对应的二进制遍历树仅包含碰撞节点(中间节

点)和叶子节点. 假设系统内存在 n 个待识别标签,那么采用 CT 算法识别,所需的总时隙为 $N_t = N_c + n$. 其中 N_c 表示碰撞节点的个数,一个碰撞节点会产生 2 个子节点,故 N_t 可以改写为 $N_t = 2 * N_c + 1$,可以得出 $N_c = n - 1$. 因此,CT 算法识别 n 个标签所需的总时隙数为:

$$N_t = 2n - 1 \tag{1}$$

表 1　QT 算法识别示例

时隙	查询前缀	标签响应数据	时隙状态
1	空串 ε	xxxxxxxx	碰撞
2	0	xxxxxxx	碰撞
3	00	xxxx1x	碰撞
4	000	01011	成功识别 A
5	001	10110	成功识别 B
6	01	100100	成功识别 E
7	1	x0x1x0x	碰撞
8	10	011000	成功识别 C
9	11	001010	成功识别 D

CT 算法所能实现的系统吞吐率为 $n/(2n-1)$,当 n 较大时(例如 $n = 1000$),其吞吐率维持在 50.03% 左右. 尽管 CT 算法消除了传统 QT 算法的空闲节点,但其还是基于二叉树查询,当碰撞较为明显时,不能加快标签识别进程.

2.3　CCMA 算法

CCMA 算法是一种多进制搜索算法,该算法引入了自定义查询命令和碰撞数据映射,碰撞数据的映射关系如表 2 所示. CCMA 算法的主要思想是[5]:如果读写器检测到第一位和第二位碰撞位是连续的,那么读写器在下一个时隙会发送一个自定义查询命令 QueryP 让碰撞的标签返回一个 4 比特映射数据,这个映射数据准确的反映了标签的碰撞信息. 读写器可以通过接收到的映射数据来准确识别标签的首 2 位碰撞信息,从而确定下一条查询命令. 如果第一位和第二位碰撞位非连续,读写器会采用 CT 算法来识别标签. 表 3 所示为采用 CCMA 算法完成对表 1 中 5 个标签识别的识别过程.

表 2　映射关系表

碰撞信息(2 bits)	映射数据(4bits)
00	0001
01	0010
10	0100
11	1000

表 3　CCMA 算法识别示例

时隙	查询前缀	标签响应数据	时隙状态
1	空串 ε	xxxxxxxx	碰撞
2	QueryP	xxxx	碰撞
3	00	xxxx1x	碰撞
4	QueryP	x00x	碰撞
5	0000	1011	成功识别 A
6	0011	0110	成功识别 B
7	01	100100	成功识别 E
8	10	011000	成功识别 C
9	11	001010	成功识别 D

从表 3 中可以看出,CCMA 算法识别 5 个标签所需的时隙数与 CT 算法相同,但是总的数据传输量要小于 CT 算法. 作者在文献[5]中仅给出了 CCMA 算法的仿真结果,为了更好的评估 QT 类算法的性能,我们对 CCMA 算法做以下性能分析.

假设系统内存在 n 个标签,采用 CCMA 算法识别,所需的总时隙数为:

$$N_t = P_{cc} * N_{cc} + P_{sc} * N_{sc} \tag{2}$$

其中 N_{cc} 为当所有的碰撞都为连续碰撞时,识别 n 个标签所需的时隙数,N_{sc} 为当所有的碰撞都为非连续碰撞(第一位和第二位碰撞位非连续)时,识别 n 个标签所需的时隙数. P_{cc} 和 P_{sc} 分别为连续碰撞和非连续碰撞出现的概率.

引理 1:多标签识别过程中产生的所有碰撞皆为非连续碰撞时,CCMA 算法识别 n 个标签所需的时隙数为 $2 * n - 1$.

证明:当所有的碰撞均为非连续碰撞时,CCMA 算法的识别过程同 CT 算法类似,此时 $N_{sc} = 2 * N_c + 1 = 2 * n - 1$. 引理 1 得证.

引理 2:识别过程中产生的所有碰撞皆为连续碰撞时,CCMA 算法识别 n 个标签所用的时隙数为 $N_{cc} = (20n - 11)/9$.

证明:当所有的碰撞均为连续碰撞时,CCMA 算法的识别过程类似一颗完全四进制遍历树. 树中的碰撞节点会产生 4 个子节点,那么有 $N_4 = N_l + N_c = 4 * N_c + 1$,其中 N_l,N_c 分别为完全四进制遍历树中的叶子节点数和碰撞节点数. 我们有:

$$N_l = 3 * N_c + 1 \tag{3}$$

在四进制遍历树中,叶子节点可能是成功节点,也有可能是空闲节点,因此式(3)可以改写为:

$$n + N_i = 3 * N_c + 1 \tag{4}$$

在 CCMA 算法中,一个连续碰撞节点可能产生 0 ~ 2 个空闲节点. CCMA 算法可以通过 QueryP 命令消除空

闲时隙,接下来,我们分别考虑这三种场景.

场景 1:当一个连续碰撞节点产生的空闲节点数为 0 时,$N_i=0$,由式(4)可以得出 $N_c=(n-1)/3$,将其代入 $N_4=4*N_c+1$ 可以求得 $N_4=(4n-1)/3$. 由 CCMA 算法原理可知,CCMA 算法中自定义命令 QueryP 的发送次数为连续碰撞产生的次数即 N_c,因此可以求出 $N_{cc}=N_4-N_i+N_c=(5n-2)/3$.

场景 2:当一个连续碰撞节点产生的空闲节点数为 1 时,$N_i=N_c$,由式(4)可以得出 $N_c=(n-1)/2$,将其代入 $N_4=4*N_c+1$ 可以求得 $N_4=2n-1$. 进一步可以求出 $N_{cc}=2n-1$.

场景 3:当一个连续碰撞节点产生的空闲节点数为 2 时,同理可以求得 $N_{cc}=3n-2$.

由于上述三种场景以等概率方式出现,所以我们可以得出 $N_{cc}=(20n-11)/9$,因此引理 2 得证.

在多标签识别过程中,连续碰撞和非连续碰撞出现的概率均为 0.5,结合引理 1 和 2,式(2)可以改写为:

$$N_t=\frac{1}{2}N_{sc}+\frac{1}{2}N_{cc}=\frac{19n-10}{9} \tag{5}$$

CCMA 算法所能实现的吞吐率为 $9n/(19n-10)$,当 n 较大时(例如 $n=1000$),其吞吐率维持在 0.4739 左右. 尽管 CCMA 算法的吞吐率低于 CT 算法,但是其所需的数据传输量低于 CT 算法.

2.4 QAS 算法

QAS 算法也采用了同 CCMA 算法类似的数据编码机制,并制定了如下编码规则:

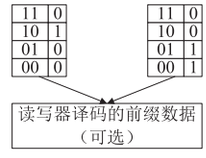
(1)假设待编码的数据长度为 $k(k=\min(\log_2(m/2),M))$,其中 m 和 M 分别为标签 ID 长度和连续碰撞的比特位数. 初始化变量 $I=0$,$S=0$ 和一个全 0 的 Q 比特数据串.

(2)$I=\sum_{j=0}^{k-1}b_j\times 2^j$

(3)令 $S=2^I$,$S^1=S|Q$,其中 S^1 为编码后的数据.

值得注意的是,当 $k=2$ 时,QAS 算法可以得到同表 2 相同的编码结果. 与 CCMA 算法不同的是,QAS 算法考虑了 $k>2$ 的情况,当连续碰撞比特数大于 2 时,QAS 算法并不会直接对 k 比特数据进行编码. 即:假设待编码数据为 $D_cD_{c-1}D_{c-2}$,编码后的数据为 $f(D_cD_{c-1})+f(D_{c-1}D_{c-2})$而不是 $f(D_cD_{c-1}D_{c-2})$,其 f 表示上述的编码函数. 在识别过程中,读写器根据接收到的编码数据译码出所需的查询前缀,将其压入堆栈,然后不断的从堆栈中提取查询前缀来发送 indexRep (L)命令. 当标签接收到 indexRep (L)命令后,会将自身的 ID 与 indexRep (L)命令中的查询前缀进行匹配,若 ID 中的高 L 位与查询前缀相等,则标签返回 ID 余下部分的高 r 位的编码数据,否则标签等待 2^r 比特数据延迟后,返回匹配前缀后的剩余 ID. 图 3 为采用 QAS 算法识别表 3 中 5 个标签的识别过程. 从图中可以看出,QAS 算法识别 5 个标签只需要 6 个时隙,相比于 CT 和 CCMA 算法,极大的缩短了识别时间.

时隙	查询前缀	标签响应数据		时隙状态
1	空串ε	xxxxxxxx		碰撞
2	Index (3)	xxxxxxxx		碰撞
3	000	0010		成功识别A
4	011	0001	10110	成功识别E、B
5	101		11000	成功识别C
6	110	0010		成功识别D



编码数据		
标签	D_7D_6	D_6D_5
A	0001	0001
B	0001	0010
C	0100	0001
D	1000	0100
E	0010	1000

图3 QAS算法识别示例

假设读写器发送初始化查询命令后,检测到最高连续碰撞比特数为 k,那么在整个识别过程中,查询深度可以计算为 $1+1+\left\lceil\frac{m-k}{2}\right\rceil$,其中 m 为标签的 ID 长度,$\lceil*\rceil$表示向上取整. 假设系统内待识别标签数为 n,采用 QAS 算法识别,所需的总时隙数为[10]:

$$T_{total}=\begin{cases}\sum_{j=0}^{1+1+\left\lceil\frac{m-k}{2}\right\rceil}N_jP_{col}^j+n, & n>1\\ 1, & n=1\end{cases} \tag{6}$$

其中 N_j 表示遍历树中第 j 层的节点数,P_{col}^j 定义为整棵查询树中,某个节点为碰撞节点的概率. 对于根节点来说,它位于遍历树中的第 0 层,因此总是可以被遍历到. 对于其他节点而言,它们被遍历到的先决条件是其父节点为碰撞节点. $P(i,j)$ 为第 j 层的某个节点有 i 个子节点的概率,P_{col}^j 和 $P(i,j)$ 可以表示为:

$$P_{col}^j=\begin{cases}\sum_{j=2}^{1+1+\left\lceil\frac{m-k}{2}\right\rceil}\sum_{i=2}^{N_j}P(i,j), & 2\le n\le 2+\left\lceil\frac{m-k}{2}\right\rceil\\ 1, & j=0,1\end{cases} \tag{7}$$

$$P(i,j)=\binom{n}{i}\frac{\binom{2^{m-k-j+2}}{i}\binom{2^m-2^{m-k-j+2}}{n-i}}{\binom{2^m}{i}\qquad\binom{2^m-i}{n-i}} \tag{8}$$

由式(6)~(8)和作者在文献[10]中的仿真结果显示,采用 QAS 算法识别 n 个标签所需的总时隙数为 $T_{total}\approx 1.5n$. QAS 算法所能实现的吞吐率稳定在 0.67 左右,明显优于 QT 和 CCMA 算法. QAS 算法的编码译码规则较 CCMA 算法复杂,这样会增加读写器和标签的设计复杂度,且当标签数较多时,在识别过程初期,读写器需要不断的译码查询前缀,造成较高的计算复杂度.

3　DPPS 算法

3.1　系统传输模型

通过对目前主流的 QT 防碰撞算法的性能分析表明,QT 防碰撞算法的识别性能还有较高的提升空间. 为了进一步提高 RFID 系统的识别性能和降低算法的实现复杂度,本节提出了一种基于双查询前缀匹配的防碰撞算法(dual prefix probe scheme,DPPS). DPPS 算法的基本原理是:为一个查询命令分配两个前缀,这两个前缀的值仅相差 1. 为了减少总的查询时隙数和避免空闲时隙,标签会将自身的 ID 和接收到的查询命令中的两个前缀相匹配. 同前缀 1 相匹配的标签返其 ID 与查询前缀相匹配后的剩余部分. 同前缀 2 相匹配的标签会延迟 r 比特的传输时间后返回其 ID 与查询前缀相匹配后的剩余部分,其中 r 等于完整的 ID 减去前缀 1 后的比特数. 若发生碰撞,读写器会根据碰撞信息更新前缀堆栈,然后继续识别碰撞标签,直到堆栈为空. 为了更直观的显示 DPPS 算法的优势,图 1 描述了读写器与标签的系统传输模型. 该系统传输模型显示了不同算法下读写器与标签双向通信时的链路时序. 该系统传输模型基于现有的超高频 RFID 工业标准 EPCglobal C1 Gen2 和 ISO 18000 - 6B. 其中 $T1$ 为读写器查询命令发送完毕和接收到标签响应之前的间隔时间,$T2$ 为读写器接收完标签响应后再次发送查询命令之前的间隔时间,$T3$ 为读写器接收完与前缀 1 匹配的 ID 数据后的等待时间. 从图 1 可以看出,尽管 DPPS 算法的单个时隙的间隔时间大于传统 QT 算法(如 CT、CCMA 算法),但是该算法那可以提高 RFID 系统的整体识别性能.

3.2　DPPS 算法原理

同传统 QT 算法类似,在 DPPS 算法中,读写器也需要一个前缀堆栈来记录和更新每个时隙所需的查询前缀. 在识别过程初始化时,读写器从堆栈获取一个初始化前缀(称为空串 ε),然后广播给其工作域内的标签. 当标签接收到这个初始化查询命令时,会返回自身的完整 ID. 一旦发生碰撞,读写器根据碰撞比特检测机制来更新堆栈,并在下一个时隙发送一个 PROBE_EQ(Com_Str,Pre1,Pre2)命令来识别标签,其中 Com_Str 表示标签 ID 最高碰撞位之前的数据部分,Pre1 表示前缀 1,Pre2 表示前缀 2. Pre2 的值等于 Pre1 的值减 1. 直到整个前缀堆栈为空后,读写器会终止整个识别过程. 图 4 给出了读写器详细的查询命令格式和标签响应的数据格式. 由于现有的超高频 RFID 工业标准 ISO 18000 - 6B 支持自定义命令,因此我们提出的 DPPS 算法可以兼容于现有标准.

PROBE_EQ 命令中 Com_Str,Pre1 和 Pre2 的配置取决于最高碰撞比特的位置. 给定一个二进制数据串

命令头	命令长度	地址	掩码	前缀	CRC-16
13 比特	8比特	8比特	8比特	1比特	16比特

(*a*) CMD_INI命令格式

前导码	ID	CRC16
9比特	96比特	16比特

(*b*) 标签对 CMD_INI 命令的响应格式

命令头	命令长度	地址	掩码	**Com_Str**	**Pre1**	**Pre2**	CRC-16
13比特	8比特	8比特	8 比特	可变	可变	**Pre1**-1	16比特

(*c*) PROBE_EQ命令格式

前导码	数据	CRC16
9比特	可变	16比特

(*d*) 标签对PROBE_EQ命令的响应格式

图4　读写器命令与标签响应格式

"$P_1P_2\cdots P_c\cdots P_k$",$k$ 为标签的 ID 长度,$P_1P_2\cdots P_{c-1}$ 为当前的查询前缀. 假设 P_c 为最高碰撞位,那么读写器会将 $P_1P_2\cdots P_{c-1}1$ 和 $P_1P_2\cdots P_{c-1}0$ 作为下一条 PROBE_EQ 命令的查询前缀,其中 $P_1P_2\cdots P_{c-1}$,1 和 0 分别为 COM_Str,Pre1 和 Pre2. 由于 DPPS 算法可以在同一个时隙内查询多个标签,因此可以极大的减少整个识别过程中读写器与标签之间的通信时间. 图 5 所示为 DPPS 算法的流程图.

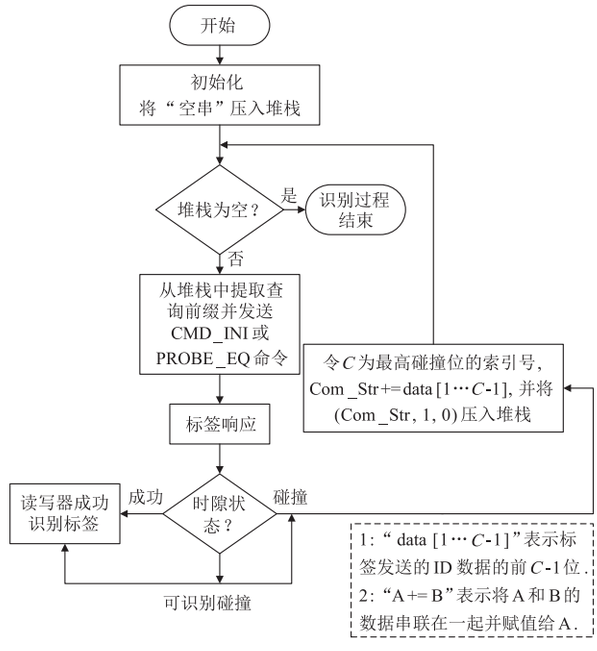


图5　DPPS算法流程图

从图 5 可以得知,DPPS 算法总共有三种不同的时隙状态:

(1)成功:在传统的 QT 类算法中,一个成功时隙意味着一个标签被读写器成功识别. 在 DPPS 算法中,一个成功时隙意味着有两个不同的标签分别与 Pre1 和

Pre2 匹配,读写器在同一个时隙中完成了对 2 个标签的识别.

(2)完全碰撞:当两个查询前缀都有多个标签与之相匹配时,会产生碰撞,导致读写器无法识别任何标签.

(3)可识别的碰撞:尽管读写器在同一时隙接收到了多个标签返回的 ID,但是能够获得一个循环冗余码校验(Cyclic redundancy check,CRC)正确的 ID. 读写器可以对拥有这个 ID 的标签进行识别.

为了直观的显示 DPPS 算法的性能,表 4 给出了 DPPS 算法识别表 3 中 5 个标签的识别过程.

表 4 DPPS 算法识别示例

时隙	查询前缀 (Com_Str,Pre1,Pre2)	标签响应数据		时隙状态
1	CMD_INI (ε)	xxxxxxxx		碰撞
2	PROBE_EQ (ε,1,0)	x0x1x0x	xxxxxxx	碰撞
3	PROBE_EQ (1,1,0)	001101	011000	成功识别 D、C
4	PROBE_EQ (0,1,0)	100100	xxxx1x	成功识别 E
5	PROBE_EQ (00,1,0)	10110	01011	成功识别 B、A

从该例中可以看出,DPPS 算法识别 5 个标签仅需 5 个时隙. 尽管 DPPS 算法的单个时隙的持续时间较 CT 算法和 CCMA 算法要长,但是 DPPS 算法节约了整个识别过程中的通信时间并减少了读写器的查询次数. 因此,DPPS 算法在识别多标签时具有明显优势.

3.3 DPPS 算法性能分析

在防碰撞算法中,识别特定数量的标签所需要的总时隙数是一个关键的指标,该指标与吞吐率紧密相关,能够决定吞吐率的高低. 吞吐率定义如下:

$$U = \frac{n}{T_{slots}} \tag{9}$$

其中 n 为待识别标签数,N_{slots}为识别 n 个标签所需的总时隙数. 特别地,在 DPPS 算法中,N_{slots}等于成功时隙数,完全碰撞时隙数和可识别的碰撞时隙数的总和. 令 $E(N_{slots})$定义为 N_{slots}的期望值,根据 DPPS 算法的原理,$E(N_{slots})$可以表示为:

$$E(N_{slots}) = 1 + N_c \tag{10}$$

其中 N_c 为识别过程 ID 碰撞产生的次数.

引理 1:假设系统内待识别标签数为 n,采用 DPPS 算法识别,期望的总时隙数 $E(N_{slots})$等于 n.

证明:DPPS 算法的识别过程类似一颗二进制遍历树,树上的每个节点对应一个时隙. 为了便于分析,在这里我们将可识别的碰撞和完全碰撞统称为碰撞节点. 显然,一个碰撞节点会产生 2 个子节点,因此遍历树的总节点数为 $2 * N_c + 1 = N_c + n$. 由此可知,$N_c = n - 1$. 根据公式(10),$E(N_{slots}) = 1 + n - 1 = n$,因此引理 1 得证.

4 仿真实验

实验环节,我们通过总时隙数,吞吐率,识别效率和通信负载(识别单个标签所需的平均数据量)四个指标来衡量 DPPS 算法的性能,并采用蒙特卡洛仿真方法将 DPPS 算法同现有的 QT 类防碰撞算法进行比较,包括:CT[4]算法,CCMA[5]算法和 QAS[10]算法. 在所有的仿真实验中,标签数在 100 到 1000 之间,调整步长为 100. 读写器与标签之间的通信速率为 40kpbs,标签 ID 为 96 比特,$T1$,$T2$,$T3$ 分别设置为 25,25 和 12.5 微秒. 值得注意的是,DPPS 算法的性能优势在这些仿真参数下具有鲁棒性,即:DPPS 算法的性能优势不会被特定的仿真参数所影响. 本文所提出的识别效率指标定义如下:

$$\eta = \frac{n \cdot T_{ID}}{N_{slots} \cdot (T_{CMD} + T_{EXT} + T_{DATA})} \tag{11}$$

$$N_{slots} = N_s + N_c + N_{ic} \tag{12}$$

其中 n 为待识别标签数,T_{ID}为标签完整 ID 传输所需的时间. N_{slots}为算法识别 n 个标签所需的总时隙数,包括成功时隙数 N_s,碰撞时隙数 N_c 和可识别的碰撞时隙数 N_{ic}. 在仿真过程中,N_s,N_c 和 N_{ic}的值可以由读写器统计得到. T_{CMD}为一个时隙内读写器查询命令所占用的时间,T_{EXT}为通信协调时间包括 $T1$,$T2$ 和 $T3$ 时间,T_{DATA}为一个时隙内标签有效数据的传输时间.

图 6 比较了不同算法识别 n 个标签所需的总时隙数,算法包括:CT 算法,CCMA 算法,QAS 算法和 DPPS 算法. 从图中可以看出,同其他算法相比,识别相同数量的标签,DPPS 算法所需的时隙数最少. 由于 DPPS 算法采用双前缀匹配技术,使得读写器能够在同一个时隙内识别两个标签,所以极大的减少了读写器的查询次数. 此外,DPPS 算法的理论分析结果与仿真结果一致.

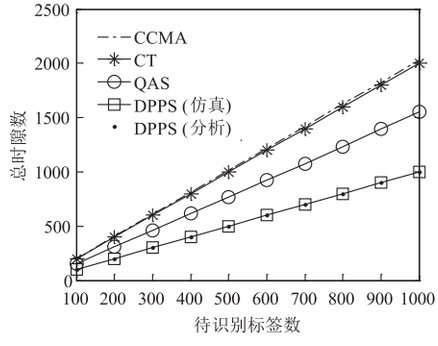


图6 仿真结果:总时隙数

图 7 描绘了不同的算法下所能实现的吞吐率. 从仿真结果中可以观测到,大部分 QT 算法的吞吐率都超过 0.5. 在众多算法中,吞吐率从高到低分别为 DPPS 算法,QAS 算法,CT 算法和 CCMA 算法. 特别地,DPPS 算法的吞吐率达到了 100%,远高于其他算法. 然而,考虑

到不同的算法之间时隙的持续时间存在差异(例如上文中讨论的,DPPS 算法的时隙持续时间要长于其他算法),算法所需的总时隙数和吞吐率皆不能很好的反映算法的时间性能. 因此,接来下我们用识别效率和通信负载来进一步衡量本文提出的防碰撞算法的性能.

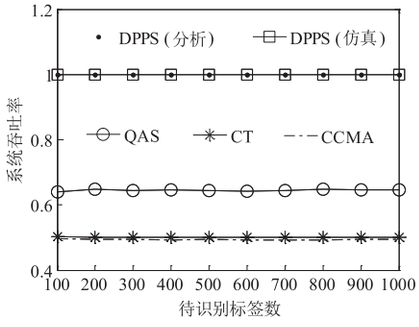


图7　仿真结果:吞吐率

图 8 显示了各个算法的时间效率,由于时间效率考虑了读写器与标签通信时的全部时间开销,因此可以很好的反应算法的时间性能. 很显然,DPPS 算法在时间效率上高于所有对比算法. DPPS 算法实现的平均时间效率为 0.4854,CT 算法,CCMA 算法和 QAS 算法实现的平均时间效率分别为 0.3443,0.4012 和 0.4427. 此外,DPPS 算法在不同的性能指标下都能保持稳定的优势. 例如,尽管 CCMA 算法识别相同数量的标签所需的总时隙数大于 CT 算法,但是 CCMA 算法的时间效率要高于 CT 算法. 原因在于,不同算法的时隙在持续时间上存在差异. 时间效率考了这种时隙间的差异,因此能够更好的反映算法的识别性能.

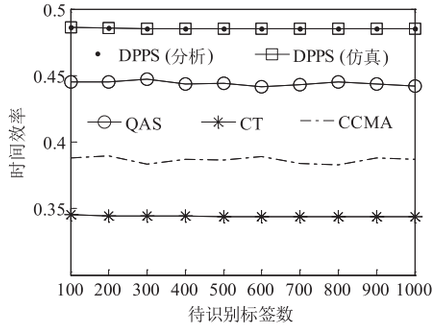


图8　仿真结果:时间效率

为了衡量算法的通信复杂度,图 9 描绘了不同算法识别单个标签所需的平均传输数据量. 尽管 DPPS 算法在单个时隙内增加了一定的数据传输,但是总体上能够维持一个较低的平均复杂度. DPPS 算法采用了图 4 所示的命令结构,避免了命令前缀的重复发送,因此可以在降低查询次数的同时,减少信息传输量,从而提高性能. 因此在相同的条件下,DPPS 算法识别一个标签所需的数据量最少仅需 247 比特,相比 QT 算法,CCMA 算法和 QAS 算法,减少了 40.9%,25.5% 和 9.31%.

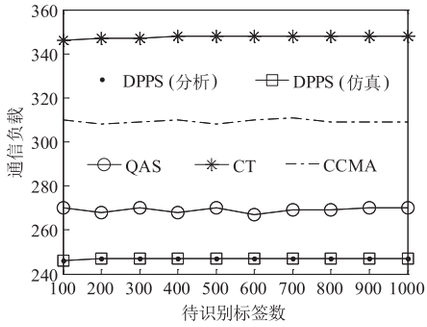


图9　仿真结果:通信负载

5　结论

本文分析了当前主流查询树防碰撞算法的性能优势和不足,在传统查询树算法的基础上提出了一种基于多前缀匹配的 DPPS 算法. 该算法能够在单个时隙内识别多个标签. DPPS 算法能够充分利用可识别的碰撞时隙来提高系统的识别效率. 为了更好的衡量 RFID 防碰撞算法的性能并让防碰撞算法更好的兼容于主流标准,本文基于现有的超高频 RFID 标准的基础上制定了读写器查询命令格式,并提出了时间效率的性能衡量指标. 理论分析和仿真结果显示,本文提出的多标签识别算法在吞吐率,时间效率和通信复杂度等方面均优于现有的查询树算法.

参考文献

[1] R Want. An introduction to RFID technology[J]. IEEE Pervasive Computing,2006,5(1):25 – 33.

[2] Duan L,Zhang X,Wang Z,Duan F. A feasible segment-by-segment aloha algorithm for RFID systems[J]. Wireless Personal Communications,2017,96:2633 – 2649.

[3] 苏健,杨晓娇,韩雨. 一种时间高效的易于实现的多标签射频识别技术[J]. 电子学报,2018,46(4):903 – 910.
Su Jian,Yang Xiao-jiao,Han Yu. A time-efficient and easy-to-implement RFID technology for multiple tags[J]. Acta Electronic Sinica,2018,46(4):903 – 910. (in Chinese)

[4] X Jia,Q. Feng,L Yu. Stability analysis of an efficient anti-collision protocol for RFID tag identification[J]. IEEE Trans Commun,2012,60(8):2285 – 2294.

[5] 苏健,文光俊,韩佳利. 一种基于 ISO 18000 – 6B 标准的 RFID 防碰撞算法[J]. 电子学报,2014,42(12):2515 – 2519.
Su Jian,Wen Guang-jun,Han Jia-li. An efficient RFID anti-collision algorithm for ISO 18000 – 6B protocol[J]. Acta Electronic Sinica,2014,42(12):2515 – 2519. (in Chinese)

[6] 王鑫,贾庆轩,高欣,陈钢,赵兵. 分组 N 叉跟踪树型 RFID 防碰撞算法研究[J]. 电子学报,2016,44(2):437

-444.
Wang Xin, Jia Qing-xuan, Gao Xin, Chen Gang, Zhao Bing. Research on grouping N-ray tracking tree RFID anti-collision algorithm[J]. Acta Electronic Sinica, 2016, 44(2):437-444. (in Chinese)

[7] Memon M, He J, Yasir M, Memon A. Improving efficiency of passive RFID tag anti-collision protocol using dynamic frame adjustment and optimal splitting[J]. Sensors, 2018, 18(4):1185-1-1184-14.

[8] Wu H, Zeng Y, Feng J, Gu Y. Binary tree slotted ALOHA for passive RFID tag anticollision[J]. IEEE Trans Parallel Distrib Syst, 2013, 24(1):19-31.

[9] SU J, WEN G, HONG D. A new RFID anti-collision algorithm based on the Q-ary search scheme[J]. Chinese Journal of Electronics, 2015, 24(4):679-683.

[10] Landaluce H, Perallos A, Zuazola I J G. A fast RFID identification protocol with low tag complexity[J]. IEEE Commun Lett, 2013, 17(9):1704-1706.

[11] Landaluce H, Perallos A, Onieva E, Arjona L, Bengtsson. An energy and identification time decreasing procedure for memoryless RFID tag anticollision protocols[J]. IEEE Transactions on Wireless Communications, 2016, 15(6):4234-4247.

[12] Landaluce H, Perallos A, Angulo I. Managing the number of tag bits transmitted in a bit-tracking RFID collision resolution protocol[J]. Sensor, 2014, 14(1):1010-1027.

[13] Jayadi R, Lai Y-C, Lin C-C. Efficient time-oriented anti-collision protocol for RFID tag identification[J]. Comput Commun, 2017, 112:141-153.

作者简介

李 川 男,1972 年出生于四川蓬溪,副教授,获四川大学硕士学位. 现为绵阳职业技术学院教师,西南科技大学控制技术四川省高校重点实验室成员. 研究方向为物联网技术、信号处理.

苏 健(通信作者) 男,1986 年生,湖北荆州人. 2008 年,2012 年和 2016 年分别在汉口学院、华中师范大学和电子科技大学获工学学士、工学硕士和工学博士学位. 现为南京信息工程大学教师,主要从事物联网技术、射频识别技术和无线网络等方面的研究工作.
E-mail:xiaoyanzi850603@163.com

刘克雄 男,1996 年出生于江西南昌,现为北京林业大学工学院在校生,主要研究方向为物联网与传感器网络.

韩 雨 男,1991 年出生于河南商丘. 现为电子科技大学通信与信息工程学院博士研究生,主要研究方向为射频集成电路与系统、无线射频识别技术等.

赵红军 男,1980 年出生于四川南充. 西南科技大学信息工程学院博士研究生. 研究方向为人工智能、图像处理.