基于信息位编码的自适应搜索 RFID 防碰撞算法研究

胡应梦,张小红
(江西理工大学信息工程学院,江西赣州 341000)

摘 要: 无线射频识别(RFID)技术可实现对目标物体的自动识别.为了减少对物体标签识别时间,提出一种基于信息位编码的自适应搜索的防碰撞(AS)算法.读写器充分利用碰撞位信息,要求标签返回碰撞位编码信息,进而自适应地生成有效查询前缀,对标签进行无空闲时隙识别,以减少查询次数,提高算法的性能.此外,AS 算法也解决了读写器与标签通信中传输信息冗余等问题.本文通过理论分析证明了该算法的有效性,其中吞吐率的理论值与实验值的误差不超过 5%,还从时间复杂度和通信复杂度对该算法进行了详细地分析.仿真结果表明:AS 算法不仅提高了系统的性能,而且还降低了标签能量的消耗.特别是当标签数为 1000 时,该算法的吞吐率仍保持在 61% 左右,比查询树算法和自适应多叉树算法的系统效率分别提高了 72% 和 20.1% 左右.

关键词: 射频识别; 防碰撞; 信息位编码; 自适应; 空闲时隙
中图分类号: TN92 **文献标识码:** A **文章编号:** 0372-2112 (2016)08-1791-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.08.003

Research of an Adaptive Searching Anti-collision Algorithm for RFID Based on Information-Bit Encoding

HU Ying-meng, ZHANG Xiao-hong
(*School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, Jiangxi* 341000, *China*)

Abstract: Radio frequency identification (RFID) technology has the ability to automatically identify the target object. An adaptive searching prefix (AS) anti-collision algorithm for RFID based on encoding is proposed to reduce the identified time of the object tag. The reader makes full use of the collision information to adaptively generate a valid query prefix by asking the tags return the collision coded information. With no idle slots for tags to identify, AS reduces the number of queries and consumedly enhances the system efficiency. Besides, it has solved the problems of redundant data in the communication between the reader and the tags and other related issues. The effectiveness of the algorithm has been proved by the theoretical analysis in detail, and the error of the throughput between the values of the theory and the experiment does not exceed 5%. Simulation results show that AS not only achieves much better performance of the system, but also reduces the energy consumption of the tags. It improves the system efficiency by 72% and 20.1% respectively compared with Query Tree algorithm and Adaptive Multi-tree Search algorithm when the number of tags is over 1000, the throughput still maintains at about 61%.

Key words: radio frequency identification (RFID); anti-collision; encoding information bit; adaptive; idle slots

1 引言

射频识别(Radio Frequency Identification, RFID)技术[1]是一种以电磁波为传输媒介的非接触式自动识别技术,是物联网感知层的关键技术之一.其基本原理是利用射频信号的空间耦合(电感或电磁耦合)或反射的传输特性来传递信息,以实现快速识别目标或数据交换的目的[2].与传统的识别技术相比,RFID 技术具有操作简单、抗干扰性强、能识别高速运动的物体以及可适应于恶劣环境等多方面的优点,在物流、跟踪、定位、工

收稿日期:2015-02-03;修回日期:2015-04-15;责任编辑:蓝红杰
基金项目:国家自然科学基金(No. 61363076, No. 11062002);江西省自然科学基金(No. 20142BAB207020);江西省教育厅科技项目((No. GJJ14465);江西省研究生创新专项资金(No. YC2014-S370)

业自动化以及交通运输控制管理等领域已得到广泛应用.因此,被列为 21 世纪最有发展前途的信息技术之一[3].

通常 RFID 系统通常由电子标签(Tag)、读写器(Reader)和后端数据库(Database)三大部分组成.其中电子标签可分为有源标签和无源标签[4],由于成本等各因素的限制,大多数 RFID 系统均采用无源标签,即被动标签,其所需能量完全由读写器发送的电磁波提供.当多个标签在同一时刻响应读写器的命令时,信号之间就会相互干扰,导致读写器无法准确地读取数据,这种现象成为标签碰撞[5].因此,如何有效快速地识别标签一直都是 RFID 技术的主要研究方向,也是物联网技术领域中一个研究的热点.

RFID 系统必须采用一定的策略或算法来避免碰撞现象的发生,即控制标签的响应时序逐个与读写器进行通信,并在一定时间内完成对所有标签的识别.为此,国内外的学者在关于多标签与读写器的碰撞问题方面已经做了大量的研究[6-8].这些防碰撞的算法主要分为两大类:一种是基于时隙随机分配的 ALOHA 算法,也称为不确定性算法;如时隙 ALOHA 算法[9]、帧时隙 ALOHA 算法[10]、动态帧时隙 ALOHA 算法[11]以及 Q 算法[12]等,这些算法均采用碰撞则退避重传的基本思想,所以其实现过程相对比较简单.但是,随着标签数目的增加,其性能急剧恶化,研究者们又相继提出增强的动态帧时隙算法[13]、分组动态帧时隙 ALOHA 算法[14]等.虽然这些算法使系统的性能有所改善,但该类算法具有一定的随机性,可能存在单个或多个标签在很长的一段时间内均无法被成功识别,即出现标签“饥饿”问题.

另外一种则是基于二进制搜索的确定型算法,其代表算法有二进制搜索(Binary Search,BS)算法[15]、动态二进制搜索(Dynamic Binary Search,DBS)算法[16]以及跳跃式动态树(Jumping and Dynamic Searching,JDS)算法[17],查询树(Query Tree,QT)算法[18],四叉树(4-ary Query Tree,4QT)算法[19]以及自适应多叉树防碰撞(Adaptive Multi-tree Search,AMS)算法[20]等.其中,BS 算法是通过多次比较,不断减少响应标签的数目,直至对剩下唯一的标签进行识别;DBS 算法则可以动态的调整读写器查询命令和标签反馈信息的 ID 长度;而 JDS 算法是基于以上两种算法的改进,不是从根节点开始重新查询,而是退到它的上一层节点即父节点继续查询,以减少算法的时间复杂度.QT 算法则对标签的要求相对比较简单,标签不需要具有任何记忆功能,从而降低了标签的制作成本.其缺点就是识别的时间较长,吞吐率很低.4QT 算法虽然减少了碰撞时隙,同时也增加了空闲时隙.而 AMS 算法则是这两种算法的结合,通过引入碰撞因子,让读写器调整分叉数以提高系统的性能.但是这些算法识别时间相对比较长,读写器与标签之间的数据通信量比较大,其吞吐率只有 50% 左右.

现有的 RFID 防碰撞算法或多或少存在一些如算法操作复杂、对标签计算能力要求高、发送数据冗余等缺点,本文在总结前人许多经典算法的基础上,提出一种基于信息位编码的自适应搜索 RFID 防碰撞算法(Adaptive Searching,AS).AS 算法利用 Manchester 特性,可以获得所有标签碰撞位的信息,并要求标签返回碰撞位编码信息,读写器收到消息后,从而生成有效查询前缀,对标签直接进行查询;标签则只需发送序列号中与前缀互补的部分.这样不仅减少了碰撞时隙,而且还将空闲时隙减少至零,同时也降低了系统的通信量,减少了对标签能量的消耗,更适合低成本的 RFID 系统.

2 AS 算法设计

2.1 基本思想

无论是 Q 算法,四叉树搜索算法还是 AMS 算法,其基本思想都是一旦检测到最高碰撞位,则按照固定的模式生成前缀,然后依次发送这些前缀对标签进行查询.由于读写器不知道标签碰撞位的具体信息,只能不断地从堆栈中取出查询前缀,盲目地对标签进行查询,就会不可避免地产生大量的空闲时隙,最终影响系统的性能.

为了让读写器获得碰撞位信息,标签需将前 4bit(后文会解释其原因)碰撞位信息进行相应的编码,如表 1 所示.读写器利用 Manchester 的特性,检测出碰撞位的信息.如果仅有 1bit 发生碰撞,则将该碰撞位分别置 0,1,生成两个新前缀,并将其压入读写器 Stack 堆栈中;反之,则将最高碰撞位之前的信息发给标签,标签收到消息后,与自身的 ID 进行对比.如果相同,则将后续 4bit(从最高碰撞位开始)进行编码,并将编码信息返回给读写器.否则,标签不应答.由于编码之后的 16bit 中只有 1bit 是 1,其他均为 0,所以无论是哪几种情况发生碰撞,读写器均能准确无误的将原碰撞信息解码出来,最后生成相应的前缀,压入堆栈中.

2.2 AS 算法约定

为了实现这个算法,定义以下三个命令来描述该算法,假设参数 M 为查询前缀,n 为检测到的最高冲突位.

(1)请求命令 Request(M):从读写器 Stack 堆栈中取出一个查询前缀 M,以广播的形式发送给标签,标签接收到该命令后,与自身的序列号进行对比,若相同,则返回应答信息;否则,不应答.

(2)返回碰撞信息命令 Call(M,n):标签接收到该命令,将自身的序列号 ID 与读写器发送的信息 M 进行

比较,若标签自身 ID 前缀部分与 M 一致,则该标签将碰撞位前四位的信息(第 n 位至第 $n-3$ 位)进行编码后返回给读写器.

(3)休眠命令 Sleep():标签接收到该指令后,将自身的序列号 ID 与接收到的信息进行比较,若两者相同,则该标签进入休眠状态,并且以后都不再应答 Request(M)指令.

表 1　AS 信息位编码规则

序号	前 4bit 碰撞位	编码后的信息
1	0000	0000000000000001
2	0001	0000000000000010
3	0010	0000000000000100
4	0011	0000000000001000
5	0100	0000000000010000
6	0101	0000000000100000
7	0110	0000000001000000
8	0111	0000000010000000
9	1000	0000000100000000
10	1001	0000001000000000
11	1010	0000010000000000
12	1011	0000100000000000
13	1100	0001000000000000
14	1101	0010000000000000
15	1110	0100000000000000
16	1111	1000000000000000

2.3　AS 算法流程

AS 算法是一种基于编码技术的防碰撞算法,首先通过 Manchester 获得碰撞位的相关信息,然后标签利用编码的原理,将编码后的碰撞位信息返回给读写器,读写器经过反编码即可获得有效查询前缀,对标签进行有效的识别,以减少查询次数,提高算法的性能. 其协议流程如图 1 所示,具体步骤如下:

步骤 1　初始化,将读写器 Stack 堆栈清空,并把空字符串 NULL 压入堆栈中,要求在其作用范围内的标签处于待命状态.

步骤 2　读写器从堆栈中取出栈首元素,并以广播的方式发送给标签,与查询前缀相匹配的标签响应,并且返回各自 ID 中与查询前缀互补的部分. 如果读写器是首次查询,则发送 Request(NULL)命令,要求标签返回其自身完整的 ID;如果堆栈为空,则跳转至步骤 7.

步骤 3　读写器收到标签返回的信息后,首先利用 Manchester 原理定位到最高碰撞位 n 的位置,然后对 n 进行判断:

(1)如果 $n<4$,则将第 n 位分别置 0,1,生成两个新的查询前缀并压入堆栈中,返回步骤 2;

(2)否则,转入步骤 4.

步骤 4　读写器根据响应标签的数量进行如下处理:

(1)如果只有一个标签响应,则直接读出该标签的数据,并发送 Sleep()命令,让其进入休眠状态,不再参与后面的查询;

(2)如果有两个或两个以上的标签,则需进一步判断:

(a)如果仅有 1bit 发生碰撞,则将相应的碰撞位分别置 0,1,生成两个前缀并压入堆栈中,返回步骤 2;

(b)如果碰撞位是 2bit 或 2bit 以上,则读写器以广播的方式发送 Call(M,n)命令,要求条件符合的标签返回其碰撞位信息,转入步骤 5;

步骤 5　标签收到 M 消息后,与自身的 ID 前缀进行对比,如果一致,则将第 n 位至第 $n-3$ 位的信息进行相应的编码,并返回给读写器;否则,标签不应答,等待下一次查询;

步骤 6　读写器收到编码信息后,通过解码即可获得标签产生碰撞的具体信息,进而生成有效查询前缀并且压入堆栈中;

步骤 7　判断堆栈是否为空,如果为空,则识别过程结束;否则,返回步骤 2,取出栈首元素,重复上述过程直至将所有标签识别完.

下面通过一个例子来说明 AS 算法识别的过程,假设有 5 个标签,其 ID 分别为 a:11110010,b:11110000,c:11010111,d:11001101,e:11001110. 识别过程如表 2 所示,读写器首先发送 Request(NULL)命令,所有标签返回其 ID,此时读写器收到的信息为 11×××××,即可得知最高碰撞位 $n=6>4$,并且是多比特发生碰

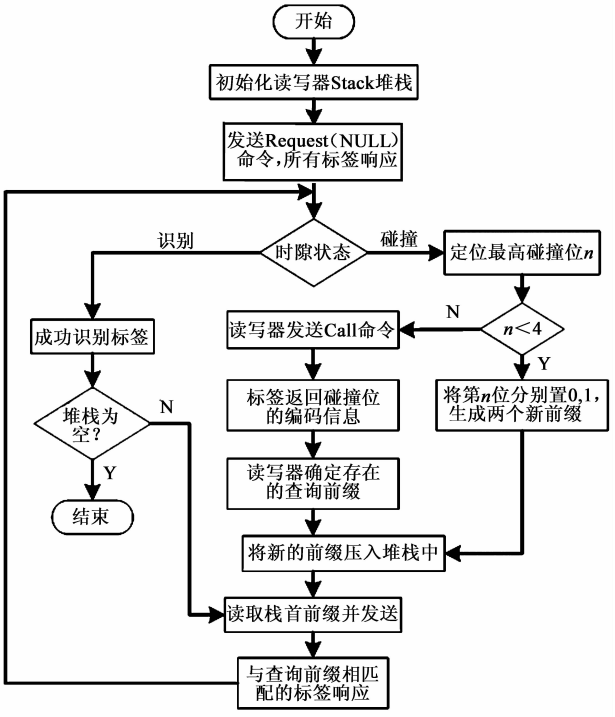

图1　AS防碰撞算法流程图

撞,所以读写器发送 Call(11,6)命令,要求前缀为 11 的标签将其 ID 上的第 6 至第 3 位的数据进行编码并返回给读写器. 然后读写器收到的碰撞反馈信息为 000×000000×0×000,由表 1,即可通过反编码获得产生碰撞的具体信息分别为 1100,0101,0011,将它们添加在前一次查询的前缀后面,形成新的查询前缀 110011,110101,111100,并将它们压入堆栈中. 在下一个时隙,读写器首先提取栈首元素 110011 发送给标签,标签的 d,e 响应,将剩余的第 1,2 位数据信息作为应答信号返回给读写器. 读写器收到信息后,即可定位到此次最高碰撞位为 $n=2<4$,将第 2 位分别置 0,1,生成两个前缀 1100110,1100111 压入堆栈中,在下一次查询中可将 d,e 分别识别出来. 同理从堆栈中取出查询前缀 110101 可直接识别出标签 c,虽然取出前缀为 111100 时,标签 a,b 产生了碰撞,但由于只有 1bit 发生碰撞,因此只需要将最高碰撞位 $n=2$ 分别置 0,1 即可识别出标签 a,b.

表 2 AS 算法举例识别过程

查询次数	查询前缀	响应标签	读写器收到信息	最高碰撞位 n	堆栈
1	NULL	a,b,c,d,e	11××××××	6	11
2	11	a,b,c,d,e	000×000000×0×000	13	110011,110101,111100
3	110011	d,e	××	2	1100110,11001101,110101,111100
4	1100110	d	1	/	11001101,110101,111100
5	1100111	e	0	/	110101,111100
6	110101	c	11	/	111100
7	111100	a,b	×0	2	1111000,1111001
8	1111000	b	0	/	1111001
9	1111001	a	0	/	∅

注:其中"×"表示碰撞,"/"表示没有,"∅"表示堆栈是空的.

3 AS 算法性能分析

AS 算法继承了 QT 算法和 AMS 算法的优点,即标签不需要存储以前的查询情况,同时在它们的基础上还加以改进. AS 算法不仅仅只有二、四两种固定分支的选择,它会随着标签 ID 的分布情况,自适应地分配多叉树,生成有效查询前缀,从而提高算法的性能. 下面,对 AS 算法的时隙情况进行分析.

3.1 多叉树的各时隙数分析

假设标签的数目为 m,L 表示是在的层数,B 为多叉树使用的叉树($B=2,4,8,\cdots$). 对于一棵满 B 叉树,有 k 个标签同时在第 L 层选择相同的节点响应,其概率服从二项分布:

$$p(k/m,L)=C_m^k p^k(1-p)^{m-k} \tag{1}$$

由此可得 m 个标签在第 L 层出现空闲的概率为:

$$p(0/m,L)=C_m^0 p^0(1-p)^m=(1-B^{-L})^m \tag{2}$$

标签成功识别的概率为:

$$p(1/m,L)=C_m^1 p(1-p)^{m-1}=mB^{-L}(1-B^{-L})^{m-1} \tag{3}$$

标签出现碰撞的概率为:

$$\begin{aligned}p(k>1/m,L)&=1-p(0/m,L)-p(1/m,L)\\&=1-(1-B^{-L})^m-mB^{-L}(1-B^{-L})^{m-1}\end{aligned} \tag{4}$$

各标签 ID 上的取值是相互独立且符合二项分布,两标签中任意一位不发生碰撞的概率为:

$$r_1=2C_2^0\left(\frac{1}{2}\right)^0\left(\frac{1}{2}\right)^2=\frac{1}{2} \tag{5}$$

两标签中任意一位发生碰撞的概率为:

$$r_2=1-r_1=\frac{1}{2} \tag{6}$$

假设标签 ID 的长度为 D,则两标签中只有 1bit 发生碰撞的概率为[21]:

$$r_3=C_D^1 r_2(r_1)^{D-1}=D2^{-D} \tag{7}$$

令 $q_{L,i/m}$ 表示在第 L 层中第 i 个节点被访问的概率,则 $q_{0,i/m}=1$,这是因为算法每次开始搜索均是从根节点出发. 除根节点外,如果要访问某个节点,其前提条件是该节点的父节点必须产生碰撞. 为了便于叙述,又设变量 $\xi_{L,i/m}$ 为在第 L 层中第 i 个节点发生碰撞的概率. 其中,

$$\xi_{L,i/m}=\xi_{L/m}=p(k>1/m,L) \tag{8}$$

$$q_{L,i/m}=q_{L/m}=\begin{cases}1, & L=0;\\ \xi_{L-1/m}, & L>0\end{cases} \tag{9}$$

因此,满 B 叉树的时隙总数应该为所有被访问节点之和,结合式(4)、(8)、(9)可得:

$$\begin{aligned}t(m)&=\sum_{L=0}^{\infty}\sum_{i=0}^{B^{L-1}}q_{L,i/m}=1+\sum_{L=1}^{\infty}B^L\xi_{L-1/m}\\&=1+B\sum_{L=0}^{\infty}B^L[1-(1-B^{-L})^m-mB^{-L}(1-B^{-L})^{m-1}]\end{aligned} \tag{10}$$

碰撞时隙数为:

$$c(m)=\sum_{L=0}^{\infty}\sum_{i=0}^{B^{L-1}}\xi_{L,i/m}=\sum_{L=0}^{\infty}B^L\xi_{L/m}=\frac{1}{B}[t(m)-1] \tag{11}$$

空闲时隙数为:

$$i(m)=t(m)-c(m)-m=\left(\frac{B-1}{B}\right)t(m)-\frac{1}{B}-m \tag{12}$$

将 $B=16$ 代入式(10),(11),(12),标签的数目由 0 增至 1000,通过 Matlab 软件仿真,如图 2 所示.

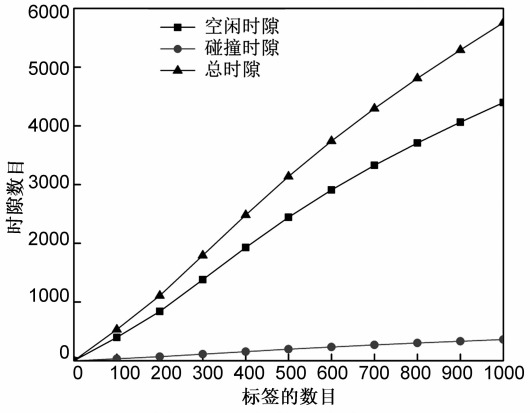

图2　16叉树的时隙情况

由图 2 可知 $t(m)$,$c(m)$,$i(m)$ 与标签数目 m 呈线性增长的关系,为了计算更为精确,本文将其拟合为二次函数曲线:

$$t(m)=-0.001m^2+6.9m-120 \tag{13}$$

$$i(m)=-0.00093m^2+5.5m-110 \tag{14}$$

$$c(m)=-6.2\times10^{-5}m^2+0.43m-7.4 \tag{15}$$

3.2　AS 算法的各时隙数分析

由于 AS 算法是采用 4bit 编码,故可认为是在 16 叉树算法上的优化. 当读写器检测到只有 1bit 发生碰撞时,读写器不需要标签返回碰撞位信息,直接将碰撞位分别置 0,1,在下一个时隙即可直接识别这两个标签. 设在识别过程中只有 1 比特发生碰撞的标签数为 $G(m)$.

$$G(m)=\sum_{L=0}^{\infty}\sum_{i=0}^{B^{L-1}}r_3=\sum_{L=0}^{\infty}B^LD\times2^{-D} \tag{16}$$

AS 算法虽然避免了空闲时隙的产生,然而,当碰撞位 $n>4$ 且为多位发生碰撞时,读写器需要发送 Call 命令以获得标签碰撞位的具体信息,进而确定存在标签之中有效前缀. 令 $F(m)$ 表示读写器发送 Call 命令的次数,则

$$\begin{aligned}F(m)&=\sum_{L=0}^{\infty}\sum_{i=0}^{B^{L-1}}p(k>1/m,L)-G(m)\\&=\sum_{L=0}^{\infty}B^L[1-(1-B^{-L})^m-mB^{-L}(1-B^{-L})^{m-1}\\&\quad-D\times2^{-D}]\end{aligned} \tag{17}$$

当读写器检测到碰撞位 $n<4$ 时,不足以进行四位编码,为了减少标签的计算复杂度,协议规定将最高碰撞位分别置 0,1,生成两个新的查询前缀. 令 $H(m)$ 为检测到碰撞位 $n<4$ 时出现的次数,由于 $H(m)$ 的情况比较复杂,本文从实验的角度对它进行分析,如图 3 所示. 可以得知 $H(m)$ 随着标签数目的增加而变化得非常缓慢,近似线性关系且所占时隙的比例很少,本文仍将其拟合为二次函数曲线:

$$H(m)=3.6\times10^{-5}m^2+0.0027m-0.17 \tag{18}$$

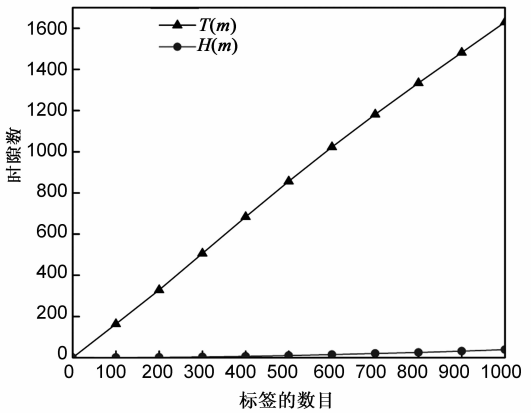

图3　两时隙数的对比

因此,AS 算法成功识别 m 个标签需要的时隙总数为:

$$T(m)=t(m)-i(m)+F(m)+H(m) \tag{19}$$

将 $B=16$,以及式(13),(14),(17)代入式(19)中,即

$$\begin{aligned}T(m)&=-0.001m^2+6.9m-120\\&\quad-(-0.00093m^2+5.5m-110)\\&\quad+\sum_{L=0}^{\infty}16^L[1-(1-16^{-L})^m-m16^{-L}(1-16^{-L})^{m-1}\\&\quad-D\times2^{-D}]+3.6\times10^{-5}m^2+0.0027m-0.17\\&=-3.4\times10^{-5}m^2+1.4027m-10.17\\&\quad+\sum_{L=0}^{\infty}16^L[1-(1-16^{-L})^m-m16^{-L}(1-16^{-L})^{m-1}\\&\quad-D\times2^{-D}]\end{aligned} \tag{20}$$

所以吞吐率为:

$$S=\frac{m}{T(m)} \tag{21}$$

4　实验仿真与分析

为了验证本算法的有效性,在 Windows 7 操作系统平台下,其中 PC 机的硬件参数为:CPU 为 i3 处理器,主频为 2.53GHZ,内存为 4G. 利用 Matlab2010a 软件作为仿真实验工具对 BS 算法,JDS 算法,QT 算法和 AMS 算法以及本文的 AS 算法进行仿真实验. 假设标签的 ID 为 16bit,并且是随机均匀分布的,标签的数目由 100 增加至 1000,仿真的结果是在相同条件下取 100 次实验的平均值.

4.1　误差分析

在第 3.2 节中本文已经对各时隙情况进行理论分析,通过实验仿真和理论计算,其分析结果如图 4 所示.

从图中可以看出吞吐率的实验值与理论计算值非常逼近,且曲线相对比较平稳. 两者的最大误差为 0.043 < 5%,所以在误差允许范围内实验结果与理论分析的结果是一致的,同时也从实验的角度验证了理论的正确性. 出现误差的原因主要有以下几个原因:

(1)标签的序列号是随机产生的,并不是理论上均匀分布,而吞吐率与序列号密切相关.

(2)在计算的过程中,有取整的运算,即有一定精度的损失.

(3)$H(m)$是通过实验统计而获得,因此也有一定的误差.

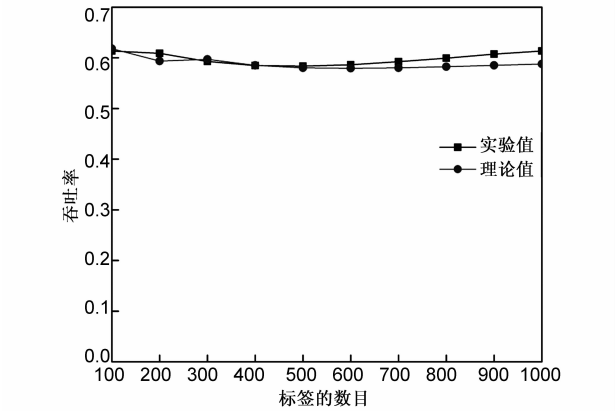

图4 吞吐率的理论值与实验值的对比

4.2 信息位编码长度的选择

读写器为了获得有效查询前缀,需要标签返回碰撞信息. 如果不将碰撞信息进行编码,则读写器很难判断各标签返回的具体信息. 本文取信息位编码长度 V 分别为 2,3,4,5bit 进行实验仿真,如图 5 所示,虽然 V 越长,吞吐率越高,但其代价成本也会随之增加. 考虑到标签成本及各方面的因素,AS 算法选择 $V=4$bit. 从图 5 中可以看出,当 $V=2$ 时,其吞吐率仅为 0.46 左右. 当 $V=3$ 时,其性能有所提高,但仍不理想. 这是由于每次标签发生多比特碰撞时,读写器为了获得碰撞位的具体信息,需要额外的发送一次 Call 命令. 当 $V=2$,3bit 时,每次查询的步长较少,碰撞较为剧烈,因此,其时隙总数也必定会增加,最终导致吞吐率较低.

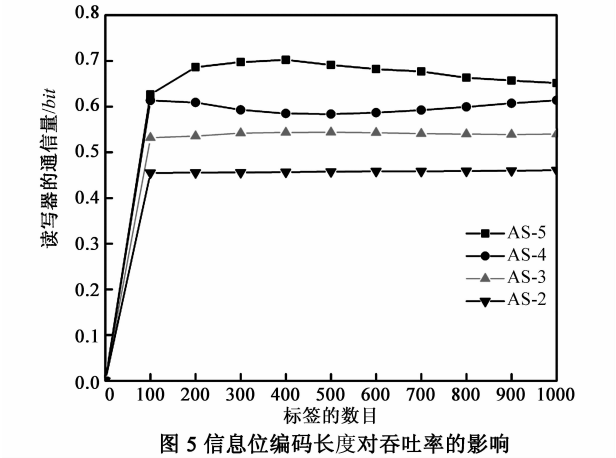

图 5 信息位编码长度对吞吐率的影响

4.3 通信复杂度分析

通信复杂度分析是评估一个算法性能的重要方法,其包括读写器通信复杂度和标签通信复杂度分析两个部分. 本文分别对 AS 算法、AMS 算法、JDS 算法以及 QT 算法在识别过程中标签和读写器的通信量进行仿真,仿真中用到的 AS 算法命令和参数长度如表 3 所示.

表 3 AS 算法中涉及的命令和参数长度

命令参数	功能	长度/bit
Request	发送查询前缀,与之相同的标签应答	22
Call	要求碰撞标签返回碰撞信息	18
Sleep	进入静默状态	18

(1)读写器的通信复杂度

读写器通信复杂度是指读写器识别全部标签所发送的总比特数,即读写器的通信量. 从图 6 可以看出,各算法中读写器的通信量都是随着标签数目的增加,呈线性增长,其中 QT 算法增长最快,AMS 算法和 JDS 算法增长速度接近,而 AS 算法增长较为缓慢. 这是因为 AS 算法能够自适应的生成有效查询前缀,大幅度地减少了读写器发送询问指令的次数. 当标签数目为 1000 时,QT 算法中读写器的通信量为 79681bit,JDS、AMS 算法分别为 61978bit 和 61098bit,而 AS 算法仅为 52697bit,比 QT 算法中读写器的通信量下降了约 33.9%,比 JDS、AMS 算法分别下降了约 15% 和 13.8%.

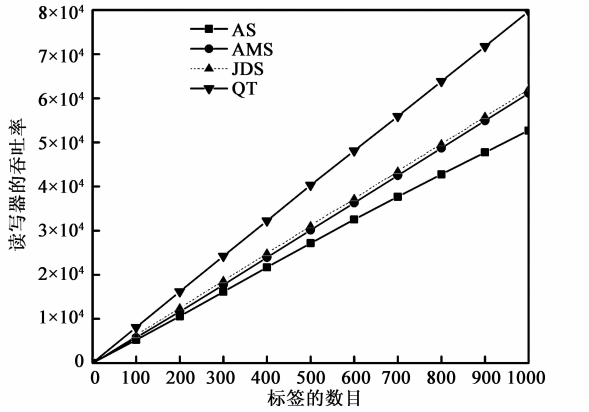

图6 读写器的通信量

(2)标签的通信复杂度

在识别的过程中,标签对于读写器的不同命令,需进行相应的响应,而所有标签发送的比特数之和称为标签的通信量. 标签的通信量与功耗密切相关,通信量越大,标签的功耗也随之增大,而对于普通的标签而言,功耗是有限的. 此外,减少标签返回的数据量,也就降低了信息泄露的危险,提高了系统的安全

性. 因此,应尽可能的减少标签的通信量. AS 算法继承了 QT 算法的优点,这不仅简化了标签的设计,也降低了标签的通信量. 如图 7 所示,在这四种算法中,AS 算法中标签的通信量增长最为缓慢,明显低于其他三种算法. 当标签的数目 1000 时,QT 算法中标签的比特数为 120700bit,JDS 算法和 AMS 算法分别为 104942bit 和 108213bit,而 AS 算法仅为 59266bit,比 QT 算法标签的通信量大约下降了 51%,比 JDS、AMS 算法分别下降了 43.5% 和 45.2%.

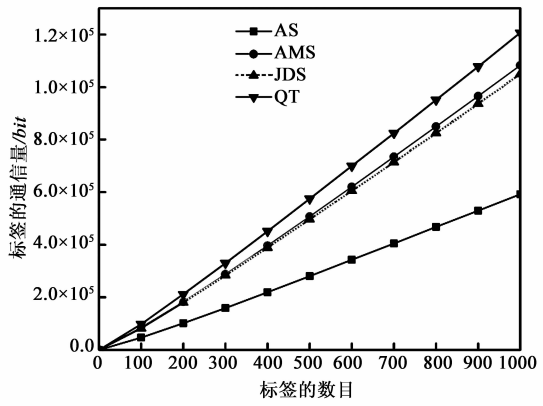

图7　标签的通信量

4.4　时间复杂度分析

读写器成功识别所有标签而需要的查询次数为时间复杂度,即总时隙数,是衡量一个算法性能的重要参数. AS 算法是一种无空闲时隙的防碰撞算法,能将碰撞标签准确无误地识别,并且还能自适应分配有效查询前缀,降低树的深度,有利于减少总时隙数. 如图 8 所示,四种算法的总时隙数随着标签数目的增加而不断增加,且呈线性增长的趋势,其中 QT 算法的总时隙数增长最快,AS 算法增长最慢. 随着标签数目的不断增大,AS 算法的优势更加明显. 当标签数目为 1000 时,AS 算法的总时隙数为 1630,比 QT 算法减少约 1174 个,比 JDS 算法减少约 369 个,比 AMS 算法减少约 327 个,其时间复杂度分别下降了 41.87%,18.46% 和 16.71%.

4.5　吞吐率分析

吞吐率也是衡量一个算法性能的一个重要指标,与总时隙数密切相关. 本文通过与 QT 算法,JDS 算法以及 AMS 算法这三种经典防碰撞算法进行比较,如图 9 所示. 这四种算法的吞吐率均保持相对比较平稳,随着标签数目的增加,QT 算法的吞吐率为 0.35 左右,JDS、AMS 算法则分别保持 0.5 和 0.51 左右,而 AS 算法的吞吐率最高,维持在 0.6 左右. 当标签的数目为 1000 时,与 QT、JDS 以及 AMS 算法相比,AS 算法的吞吐率分别提高了 72%,22.7% 和 20.1%.

5　结论

本文提出一种基于信息位编码的自适应搜索 RFID 防碰撞(AS)算法,同 QT 算法一样,标签不需要记忆以前的查询情况,只需要与读写器发出的前缀作比较. 若匹配成功,则仅需发送前缀后面的序列以减少标签的数据传输.

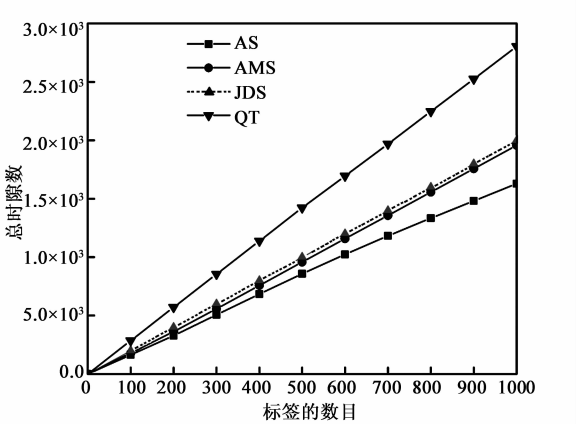

图8　四种算法的总时隙数对比

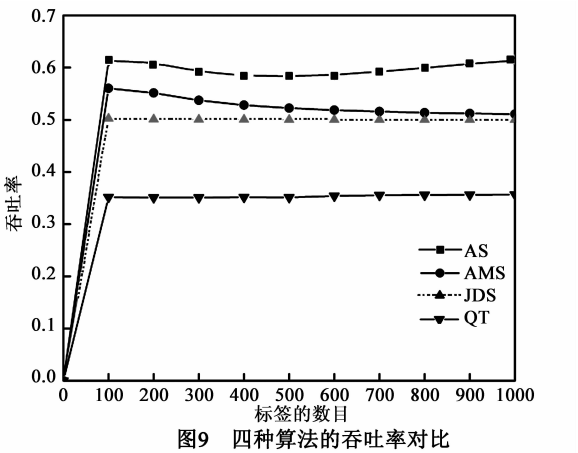

图9　四种算法的吞吐率对比

本文虽然是在选取标签数目为 1000 以内,对 AS 算法进行的实验仿真和理论分析,但通过实验表明和数据分析,当标签数目为 1200,1500 时,除了误差有稍微的增加外,上述大部分结论也依然成立. 此外,还从时间复杂度和通信复杂度对该算法进行了详细地分析. 其仿真结果表明,随着标签数目的增加,AS 算法同 QT、JDS 以及 AMS 算法一样,标签、读写器的通信量以及时隙总数几乎都保持了线性增加. 其中,AS 算法增长最为缓慢,通信量和时隙总数最少,而吞吐率最高,维持在 60% 左右. 因此,本文提出的 AS 算法能有效地提高 RFID 系统的工作效率,减少识别时间,降低标签的能量消耗. 针对于大量的标签的识别,AS 算法的优势尤为显著,在物联网工程中具有广泛的应用前景.

参考文献

[1] Atzori L, Iera A, Morabito G. The internet of things: A survey[J]. Computer Networks, 2010, 54(15): 2787 – 2805.

[2] Want R. An introduction to RFID technology[J]. IEEE Pervasive Computing,2006,5(1):25 - 33.
[3] Weinstein R. RFID:A technical overview and its application to the enterprise[J]. IT Professional,2005,7(3):27 - 33.
[4] Yang L,Rida A,Vyas R,et al. RFID tag and RF structures on a paper substrate using inkjet-printing technology[J]. IEEE Transactions on Microwave Theory and Techniques,2007,55(12):2894 - 2901.
[5] Hwang K,Yeo S S,Park J H. Distributed tag access with collision-avoidance among mobile RFID readers[A]. Proceedings of the 2009 International Conference on Computational Science and Engineering[C]. Vancouver:IEEE,2009. 621 - 626.
[6] Qian C,Ngan H,Liu Y,et al. Cardinality estimation for large-scale RFID systems[J]. IEEE Transactions on Parallel and Distributed Systems,2011,22(9):1441 - 1454.
[7] Wu H,Zeng Y,Feng J,et al. Binary tree slotted ALOHA for passive RFID tag anti-collision[J]. IEEE Transactions on Parallel and Distributed Systems,2013,24(1):19 - 31.
[8] Li Z,He C,Li J,et al. RFID reader anti-collision algorithm using adaptive hierarchical artificial immune system[J]. Expert Systems with Applications,2014,41(5):2126 - 2133.
[9] Liu L,Lai S. ALOHA-based anti-collision algorithms used in RFID system[A]. Proceedings of the 2006 International Conference on Wireless Communications,Networking and Mobile Computing[C]. Wuhan:IEEE,2006. 1 - 4.
[10] Eom J B,Lee T J. Accurate tag estimation for dynamic framed-slotted ALOHA in RFID systems[J]. IEEE Communications Letters,2010,14(1):60 - 62.
[11] Deng D J,Tsao H W. Optimal dynamic framed slotted ALOHA based anti-collision algorithm for RFID systems[J]. Wireless Personal Communications,2011,59(1):109 - 122.
[12] Maguire Y,Pappu R. An optimal Q-algorithm for the ISO 18000-6C RFID protocol[J]. IEEE Transactions on Automation Science and Engineering,2009,6(1):16 - 24.
[13] Chen W T. An accurate tag estimate method for improving the performance of an RFID anti-collision algorithm based on dynamic frame length ALOHA[J]. IEEE Transactions on Automation Science and Engineering,2009,6(1):9 - 15.
[14] Wang C Y,Lee CC. A grouping-based dynamic framed slotted ALOHA anti-collision method with fine groups in RFID systems[A]. Proceedings of the 2010 5th International Conference on Future Information Technology[C]. Busan:IEEE,2010. 1 - 5.
[15] Bentley J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM,1975,18(9):509 - 517.
[16] Yu Z,Liu X. Improvement of Dynamic Binary Search Algorithm Used in RFID System[A]. Proceedings of the 2011 International Conference on Cross Strait Quad-Regional Radio Science and Wireless Technology[C]. Harbin:IEEE,2011. 1046 - 1049.
[17] Yu S,Zhan Y,Wang Z,et al. Anti-collision algorithm based on jumping and dynamic searching and its analysis[J]. Computer Engineering,2005,31(9):19 - 20.
[18] Choi J H,Lee D,Lee H. Query tree-based reservation for efficient RFID tag anti-collision[J]. IEEE Communications Letters,2007,11(1):85 - 87.
[19] Shih B Y,Lo T W,Chen C Y. The research of quadtree search algorithms for anti-collision in radio frequency identification systems[J]. Scientific Research and Essays,2011,6(25):5342 - 5350.
[20] 丁治国,朱学永,郭立,等. 自适应多叉树防碰撞算法研究[J]. 自动化学报,2010,36(2):237 - 241.
Ding Zhi-guo,Zhu Xue-yong,Guo Li,et al. An adaptive anti-collision algorithm based on multi-tree search[J]. Acta Automatica Sinica,2010,36(2):237 - 241. (in Chinese)
[21] 刘子龙,纪金水,刘彩虹,等. 基于连续碰撞位探测的防碰撞算法研究[J]. 电子学报,2013,41(11):2156 - 2160.
Liu Zi-long,Ji Jin-shui,Liu Cai-hong,et al. An anti-collision algorithm based on continuous collision bit detection[J]. Acta Electronica Sinica,2013,41(11):2156 - 2160. (in Chinese)

作者简介

胡应梦 男,1989 年 11 月出生,湖南娄底人,现为江西理工大学信息工程学院硕士研究生,研究方向为 RFID 防碰撞算法、可信认证协议.
E-mail:huyingmeng89@163.com

张小红 女,1966 年 8 月出生,河北昌黎人,现为江西理工大学信息工程学院教授、博士生导师,研究方向为无线传感器网络、非线性动力学理论、混沌保密通信.
E-mail:xiaohongzh@263.net