

RFID 防碰撞时隙应变协处理算法研究

张小红, 张留洋

(江西理工大学信息工程学院, 江西赣州 341000)

摘 要: 射频识别(Radio Frequency Identification, RFID)技术具有同时识别多个标签的能力. 由于标签随机占用信道会使响应信号发生混叠, 当大量的标签在读写器感应区域内同时响应势必会引起标签碰撞问题, 严重影响了 RFID 的系统吞吐率. 为进一步提高 RFID 系统吞吐率, 在不增加标签端算法复杂度和硬件的条件下, 本文提出了预设调整盘存周期的差分门限、时隙前向检测后向预测和二进制散列的方法, 协同处理当前时隙和下一个时隙的标签响应情况. 利用预设差分门限的方式避免读写器进行浮点运算, 同时通过时隙前向检测和后向预测的方法提前判别当前时隙和下一个时隙的标签响应情况, 然后快速跳过当前空闲时隙和下一个空闲时隙, 并对在当前碰撞时隙和下一个碰撞时隙响应的标签实时进行二进制散列. 通过盘存周期调整方式比较、时隙散列方式对比、识别时延与吞吐率比较三个仿真实验, 结果显示改进的算法提高了读写器的计算效率, 明显减少了无效时隙占用信道的时间, RFID 系统吞吐率可提高到 52% 以上.

关键词: 射频识别; 防碰撞算法; 差分门限; 时隙前向检测; 时隙后向预测

中图分类号: TN911.23

文献标识码: A

文章编号: 0372-2112 (2014) 06-1139-08

电子学报 URL: <http://www.ejournal.org.cn>

DOI: 10.3969/j.issn.0372-2112.2014.06.016

Research on RFID Anti-Collision Algorithm of Slot Responding in Real-Time and Co-Processing

ZHANG Xiao-hong, ZHANG Liu-yang

(School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, Jiangxi 341000, China)

Abstract: Although RFID(Radio Frequency Identification, RFID) is capable of identifying multiple tags synchronously, the tag collision occurs inevitably when a large number of tags respond in the induction area of reader simultaneously, which seriously influences the throughput of RFID system because signal aliasing arises from tags occupying the channel randomly. In order to further improve the throughput of RFID system, this paper proposes a new anti-collision algorithm with adjusting inventory cycle by the preset difference threshold, the slot checking forward, the slot predicting backward and the slot binary splitting, which are coordinately combined to deal with the response of the current slot and the next slot when algorithm complexity and hardware are not added in tags. In the algorithm, the preset difference threshold can prevent reader from floating point computation. Meanwhile, the methods of slot checking forward and predicting backward are joined together to judge the response of the current slot and the next slot in advance, which can skip the current idle slot and the next idle slot, and split the tags responding in the current collision slot and the next collision slot in real time. In the following three simulation experiments, ways of inventory cycle adjustment, methods of slot splitting, the identification delay and throughput are compared respectively. The results indicate that the improved anti-collision algorithm can raise the computation efficiency of reader, effectively reduce the time of channel occupied by invalid slots, and as a result, the throughput of RFID system can reach more than 52%.

Key words: RFID; anti-collision algorithm; difference threshold; slot checking forward; slot predicting backward

1 引言

射频识别(RFID)是一种利用射频信号和空间耦合的传输特性进行双向通信, 实现对物体的非接触式自动

识别技术^[1]. 与传统条码技术相比, RFID 技术具有存储容量大、寿命长、安全性高、非视距读写、识别速度快等特点. RFID 技术作为物联网的重要支撑技术之一, 是目前最具发展前景和变革力的高新技术^[2,3].

在 RFID 系统中,读写器具有多标签识别能力.由于标签应答时随机占用信道,致使读写器不能正确获取标签的 ID,这称为 RFID 系统中的标签冲突或碰撞. 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,控制标签逐个通过射频信道与读写器通信^[4].防碰撞算法主要解决如何快速和准确地从多个标签中选出一个与读写器进行通信,并在尽量短的时间内完成对所有标签的识别.目前主要采用时分多路技术解决标签碰撞问题,基于时分多路技术的标签防碰撞算法一般分为确定性算法和非确定性算法.确定性算法,常用的确定性算法如二进制搜索算法、动态二进制搜索算法、查询树算法等^[5,6].非确定性算法,也称为概率性算法,是基于 Aloha 算法的一类重要的防碰撞算法^[7].常用的非确定性防碰撞算法有纯 Aloha 算法、时隙 Aloha 算法、帧时隙 Aloha 算法^[8,9](Frame-Slotted Aloha algorithm, FSA)、动态帧时隙 Aloha 算法^[10,11](Dynamic Frame-Slotted Aloha algorithm, DFSA)等.

EPCglobal 提出的 EPC Class-1 Generation-2 协议标准中包含一种基于 Q 值调整的 SA(Slotted Aloha, SA) 防碰撞算法,其中 Q 值用于调整读写器盘存周期.这种算法在读写器端通过 Query、QueryAdjust、QueryRep 等命令查询调整一个盘存周期内的标签.本文针对协议中 Q 值调整时缺乏灵便性和对无效时隙处理上的缺点,提出了基于差分门限调整和时隙前向检测后向预测二进制散列机制的 SA 算法——FBBS(Slotted Aloha algorithm with difference threshold and slot Forward-checking Backward-predicting Binary Splitting, FBBS).仿真结果显示,FBBS 算法能够在标签规模较大时保持较高的系统吞吐率.与已有算法相比,该算法在不增加读写器和标签任何硬件、不增加标签端算法复杂度的情况下较大地提高了系统吞吐率,降低了识别时延,FBBS 算法的系统吞吐率可稳定在 52% 以上.

2 EPC Class-1 Generation-2 防碰撞算法

EPC Class-1 Generation-2 防碰撞算法(以下简称 SA 算法)相比于其他防碰撞算法的优势在于,SA 算法允许读写器读写范围内存在多达 $2^Q - 1$ ($Q_{\max} = 15$, 最大标签数为 32767)个标签,其搜索效率可以保持在一个较为稳定的水平,不会随着标签数量的变化出现较大波动.

2.1 EPC Class-1 Generation-2 防碰撞算法中的常用指令

SA 算法主要通过盘存操作对标签进行读取.盘存操作是 SA 算法的核心操作,它包含一系列关键的指令:Query、QueryAdjust、QueryRep 和 ACK. SA 算法和 FSA 算法有类似之处,但在 SA 算法中没有帧的概念,取而代之的是盘存周期,即为两个连续的 Query 指令之间的

时间间隔.

Query 指令:是 SA 算法的初始化指令.这个指令包含一个参数 Q , Q 取 0 到 15 之间的整数.每一个收到 Query 指令的标签在 $[0, 2^Q - 1]$ 中选择一个随机数,并将其载入自身的时隙计数器(Slot Counter, SC).只有 $SC = 0$ 的标签立即在当前时隙响应,向读写器返回自身的 16 位伪随机数 RN16.若在当前时隙上只有一个 $SC = 0$ 的标签响应,读写器向该标签回复 ACK 指令确认已正确接收其 RN16,接下来进行双向认证和识别标签 ID,识别完成后该标签不再响应此读写器的查询指令;若当前时隙上没有标签响应,则为空闲时隙;若当前时隙上有多个标签响应,标签的返回信息会在读写器端混叠,此时该时隙为碰撞时隙.

QueryAdjust 指令:是 SA 算法的盘存周期调整指令.若盘存周期内出现了较多空闲时隙或碰撞时隙,为保证系统吞吐率的稳定性,此时读写器通过该指令进行调整.指令中包含参数 Q ,所有接收到该指令的标签重新在 $[0, 2^Q - 1]$ 中选择一个随机数,并将其载入 SC.

QueryRep 指令:是 SA 算法开启下一个时隙的指令.标签接收到该指令后,若 $SC = 0$,表示该标签在上个时隙发生了碰撞,将其 SC 的值修改为 7FFFH;若 $SC = 1$,则将其 SC 的值减 1 并立即向读写器返回其 RN16;若 $SC > 1$,则将其 SC 的值减 1,等待下一个 QueryRep 指令或 QueryAdjust 指令.

ACK 指令:是读写器正确收到标签返回的随机数 RN16 后,向标签发送的确认信息,用于进一步与标签进行认证.

2.2 EPC Class-1 Generation-2 防碰撞算法中 Q 值调整方案

SA 算法在执行过程中需要不断发送 QueryAdjust 指令调整标签所选择的时隙,以减少空闲时隙和碰撞时隙.是否需要发送 QueryAdjust 指令是由 Q 值是否发生变化决定的,而 Q 值的调整由当前时隙的状态(空闲、碰撞或成功)决定, Q 值的调整步长由参数 C 决定. Q 值调整方案步骤如下:

① Q_{fp} 的初始值为 4.0,读写器向识别范围内的标签发送 Query 指令.

② Q_{fp} 根据当前时隙上响应的标签数进行调整.如果当前时隙上无标签响应,则 $Q_{fp} = \max(0, Q_{fp} - C)$;如果当前时隙上只有一个标签响应, Q_{fp} 不变;如果当前时隙上多于一个标签响应,则 $Q_{fp} = \min(15, Q_{fp} + C)$.

③ Q_{fp} 以时隙为周期进行更新,读写器对 Q_{fp} 进行四舍五入得到 Q 值.

④ 若 Q 值发生变化,读写器发送 QueryAdjust 指令,所有未被识别的标签重置 SC 的值;否则读写器发送

QueryRep 指令,开启下一个时隙识别后续标签。

由此可见,该防碰撞协议通过参数 C 来统计空闲时隙和碰撞时隙并调整 Q 值,使系统吞吐率保持一个较稳定的水平。协议中推荐 C 为 $(0.1, 0.5)$ 之间的实数,但协议中没有明确给出不同 Q 值下 C 值如何选择;协议中涉及实数运算,标签规模较大时读写器浮点运算负担较大;协议对于发生碰撞的时隙并没有充分利用其碰撞信息,此外对于空闲时隙也没有做任何处理。针对协议中存在的缺陷,本文提出了 FBBS 算法,利用时隙前向检测、后向预测和二进制散列的方法对空闲时隙和碰撞时隙进行处理,弥补了协议不能有效处理空闲时隙和碰撞时隙的不足,仿真实验结果显示,FBBS 算法能大幅提高系统的吞吐率。

3 Q 值差分门限调整机制

FBBS 算法首先提出了一种基于差分门限的 Q 值调整方案,这种方案相比协议中的 Q 值调整方案更加简单轻便,且可以避免读写器进行浮点运算。

在 SA 算法中, C 值推荐在 $(0.1, 0.5)$ 内选择,且根据 Q 值动态取值,具体来说,由于 Q 每自加或自减一次,盘存周期内的时隙数会随之加倍或减半,所以当 Q

值较大时, C 应取较小的值;而当 Q 值较小时, C 应取较大的值。对应每个 Q 值, C 具体应该取何值,经验的方法是根据式:

$$C = \frac{0.8}{Q}$$

(1)

由 SA 算法中 Q 值的调整方案可知,通过参数 C 的累加控制 Q 值。本文定义 Q 值调整差分门限为 $D(Q)$, $D(Q)$ 表示以 Q 为自变量的函数 D ,在差分门限调整机制中,当盘存周期内空闲时隙数 N_{idle} 和碰撞时隙数 N_{coll} 满足式:

$$|N_{coll} - N_{idle}| = D(Q)$$

(2)

说明此时系统没有工作在最佳状态,需要对 Q 值进行调整,调整方法如式:

$$Q_{new} = Q_{old} + \frac{N_{coll} - N_{idle}}{|N_{coll} - N_{idle}|}$$

(3)

由式(1), $Q = 1$ 时, $C = 0.8$, Q 值从 1 变化到 2 时, $Q_{fp} = \min(15, Q + n \times C)$, 此时 $n = 1$; Q 值从 1 变化到 0 时, $Q_{fp} = \max(Q - n \times C, 0)$, 此时也是 $n = 1$ 。也即此时的差分门限 $D(Q) = n = 1$ 。根据这种算法,可以得出 $Q \in [1, 15]$ 时, Q 值调整差分门限 $D(Q)$ 的取值如表 1。

表 1 Q 值调整差分门限 $D(Q)$ 的取值

Q	1	2	3	4	5
C	0.800	0.400	0.267	0.200	0.160
Q_{fp}	0.2 = 1 - 1 * 0.8 或 1.8 = 1 + 1 * 0.8	1.2 = 2 - 2 * 0.4 或 2.8 = 2 + 2 * 0.4	2.47 = 3 - 2 * 0.267 或 3.53 = 3 + 2 * 0.267	3.4 = 4 - 3 * 0.2 或 4.6 = 4 + 3 * 0.2	4.36 = 5 - 4 * 0.16 或 5.64 = 5 + 4 * 0.16
$D(Q)$	1	2	2	3	4
Q	6	7	8	9	10
C	0.133	0.114	0.100	0.089	0.080
Q_{fp}	5.47 = 6 - 4 * 0.133 或 6.53 = 6 + 4 * 0.133	6.43 = 7 - 5 * 0.114 或 7.57 = 7 + 5 * 0.114	7.5 = 8 - 5 * 0.1 或 8.5 = 8 + 5 * 0.1	8.46 = 9 - 6 * 0.089 或 9.54 = 9 + 6 * 0.089	9.44 = 10 - 7 * 0.08 或 10.56 = 10 + 7 * 0.08
$D(Q)$	4	5	5	6	7
Q	11	12	13	14	15
C	0.073	0.067	0.062	0.057	0.053
Q_{fp}	10.49 = 11 - 7 * 0.073 或 11.51 = 11 + 7 * 0.073	11.46 = 12 - 8 * 0.067 或 12.54 = 12 + 8 * 0.067	12.44 = 13 - 9 * 0.062 或 13.56 = 13 + 9 * 0.062	13.49 = 14 - 9 * 0.057 或 14.51 = 14 + 9 * 0.057	14.47 = 15 - 10 * 0.053 或 15.53 = 15 + 10 * 0.053
$D(Q)$	7	8	9	9	10

这样在标签识别过程中,读写器只需统计盘存周期内空闲时隙数和碰撞时隙数,当空闲时隙数和碰撞时隙数的差分满足式(2)时,利用式(3)对 Q 值进行调整。这样做可以避免读写器在处理完一个时隙后进行浮点运算,提高读写器的计算效率和反应速度,进而降低识别时延。

4 盘存周期内时隙处理方式

4.1 二进制散列机制 BS

当某个时隙发生了碰撞,若能有效利用碰撞信息,对碰撞时隙进行及时处理,避免发生碰撞的标签与未识别标签混合,可进一步提高系统的吞吐率。在二进制散列机制中碰撞发生后,并不是简单地利用 QueryRep

指令开启下一个时隙,而是让在当前时隙上发生碰撞的标签将其时隙计数器 SC 随机置 0 或置 1,盘存周期内未识别标签 SC 值加 1^[12,13].具体调整方法如式:

$$SC = \begin{cases} \text{rand}(0/1), & SC = 0 \\ SC + 1, & SC > 0 \end{cases} \quad (4)$$

图 1 为 EPC Class-1 Generation-2 协议中 SA 防碰撞算法在 BS 机制下的标签识别过程,盘存周期调整方式采用第 3 节的差分门限调整方式.图中实例 $Q=2$,标签数为 5.在时隙数为 1 时,假设这 5 个标签 SC 的初始值设定为 1、1、0、2、4.

时隙数	1	2	3	4	5	6	7	8	9	10	11
标签1	1	0	1	0	1	0					
标签2	1	0	1	0	0						
标签3	0										
标签4	2	1	2	1	2	1	0				
标签5	4	3	4	3	4	3	2	1	0		
时隙状态	S	C	I	C	S	S	S	I	S	I	I

数字 时隙计数器

C 碰撞时隙

S 成功时隙

I 空闲时隙

图1 BS工作实例

4.2 时隙后向预测二进制散列机制 BPBS

时隙后向预测二进制散列机制 BPBS (Backward-Predicting Binary Splitting, BPBS),与时隙预测二进制选择机制 SPBS (Slot-Predicting Binary Selection, SPBS) 原理相同^[13],是在时隙二进制散列机制的基础上提出来的.在 BPBS 机制中,标签的响应数据有两部分组成^[13].响应数据格式如式(5)所示:

$$[RN16|_{SC=0}]_{16} + [RN16|_{SC=1}]_k \quad (5)$$

其中 $[RN16|_{SC=0}]_{16}$ 是 $SC=0$ 的标签为与读写器建立通信向读写器发送的 16 位随机数; $[RN16|_{SC=1}]_k$ 是 $SC=1$ 的标签返回的 RN16 前 k ($1 \leq k \leq 16$) 比特,定义为预测后缀.读写器首先处理 $[RN16|_{SC=0}]_{16}$,若发生碰撞,对当前时隙进行二进制散列;若未发生碰撞,则预测后缀有效,结合 $[RN16|_{SC=0}]_{16}$ 的情况处理 $[RN16|_{SC=1}]_k$.若预测后缀为空,所有标签 SC 减 2 以跳过下一个空闲时隙;若预测后缀唯一,则所有标签 SC 减 1 直接开启下一个时隙;否则提前对下一个时隙进行二进制散列.

图 2 为 EPC Class-1 Generation-2 协议中 SA 防碰撞算法在 BPBS 机制下的标签识别过程.盘存周期调整方式采用第 3 节的差分门限调整方式,图中实例 $Q=2$,标签数为 5.在时隙数为 1 时,假设这 5 个标签 SC 的初始值设定为 1、1、0、2、4.与图 1BS 机制比较,识别完这 5 个标签所用时隙数明显减少.

由图 2 可知,BPBS 机制可以对当前碰撞时隙进行

散列处理,避免已发生碰撞的标签与未被识别的标签再混合,降低这些标签再次发生碰撞的概率;同时可以预测到下一个时隙的响应情况,从而提前对下一个时隙进行处理,能够跳过下一个空闲时隙,提前对下一个碰撞时隙进行二进制散列处理.BPBS 机制可以较为有效地减少空闲时隙和碰撞时隙的出现,但同时也应注意到 BPBS 机制未对当前空闲时隙做任何处理.

时隙数	1	2	3	4	5	6	7
标签1	1	0					
标签2	1	1	0				
标签3	0						
标签4	2	2	1	0			
标签5	4	4	3	2	0		
时隙状态	S	S	S	S	S	I	I

数字 时隙计数器

C 碰撞时隙

S 成功时隙

I 空闲时隙

图2 BPBS工作实例

4.3 时隙前向检测后向预测二进制散列机制 FCBPBS

为进一步减少碰撞时隙和空闲时隙,提高系统吞吐率,同时针对 BPBS 机制中存在不能有效处理空闲时隙的不足,本文提出了时隙前向检测后向预测二进制散列机制 FCBPBS (Forward-Checking Backward-Predicting Binary Splitting, FCBPBS).FCBPBS 机制针对 BPBS 机制中不能对当前空闲时隙处理的不足,引入时隙前向检测机制,利用较少的数据传输开销提前判断当前时隙的标签响应情况.标签的响应数据由三部分组成.响应数据格式如式(6)所示:

$$[Check = 10]_2 + [RN16|_{SC=0}]_{16} + [RN16|_{SC=1}]_k \quad (6)$$

$[Check = 10]_2$ 为 2 比特碰撞检测位,用于检测当前时隙上是否有标签响应; $[RN16|_{SC=0}]_{16}$ 和 $[RN16|_{SC=1}]_k$ 分别是 $SC=0$ 和 $SC=1$ 的标签向读写器发送的 16 位随机数和 k 位预测后缀.

读写器首先处理 $[Check = 10]_2$,检测碰撞检测位,若不能检测到碰撞检测位,则说明当前时隙为空闲时隙,发送 QueryRep 指令立即跳过该空闲时隙并开启下一个时隙;若能,则说明当前时隙为成功时隙或碰撞时隙,然后处理 $[RN16|_{SC=0}]_{16}$ 部分,根据该部分数据是否发生碰撞进行不同的后续处理.

若经过处理 $[RN16|_{SC=0}]_{16}$ 判断当前时隙是碰撞时隙,则此时通过式(4)对该时隙进行二进制散列,让当前时隙上发生碰撞的标签将其时隙计数器 SC 随机置 0 或置 1,盘存周期内其他未被识别的标签 SC 值加 1.若经过处理 $[RN16|_{SC=0}]_{16}$ 判断当前时隙是成功时隙,读

写器此时读取在当前时隙上响应的这个标签并使其静默,该标签此后不再响应读写器的 Query 指令。

如果可以检测到碰撞检测位,并且当前时隙为成功时隙,此时读写器接着处理预测后缀部分 $[RN16]_{SC=1}^k$,除此之外其他情况下均不对预测后缀部分进行处理。若预测后缀为空,可以判定下一个时隙一定为空闲时隙,所有未被识别的标签 SC 减 2 以跳过该空闲时隙;若预测后缀发生了碰撞,可以判定下一个时隙一定为碰撞时隙。为了避免碰撞,需要将在下一个时隙响应的标签进行二进制散列,该盘存周期内其他未被识别的标签 SC 值保持不变^[13],具体如式(7):

$$SC = \begin{cases} \text{rand}(0/1), & SC = 0 \\ SC, & SC > 0 \end{cases} \quad (7)$$

若预测后缀不为空,也没有发生碰撞,此时读写器按照下一个时隙是成功时隙进行处理,所有标签 SC 减 1 直接开启下一个时隙。不过下一个时隙也可能是碰撞时隙,原因是此时预测后缀由于位数有限可能都相同,这是预测失败的情况,预测失败会造成下一个时隙发生碰撞。假设随机数发生器以相等的概率随机产生 0 和 1,预测后缀的位数为 k ,发送预测后缀的标签数为 r 。当 $k = 1$ 时,根据二项式分布定理, r 个标签的预测后缀都为 0 或 1 的概率为:

$$P = \binom{r}{r} \left(\frac{1}{2}\right)^r \left(\frac{1}{2}\right)^0 + \binom{r}{r} \left(\frac{1}{2}\right)^0 \left(\frac{1}{2}\right)^r = \left(\frac{1}{2}\right)^{r-1} \quad (8)$$

所以 r 个标签其 k 位预测后缀都相同的概率,即实际是碰撞时隙,但预测为成功时隙也即预测失败概率为:

$$P_{\text{coll-fail}} = \binom{k}{k} P^k (1-P)^0 = \left(\frac{1}{2}\right)^{(r-1)k} \quad (9)$$

则能成功预测到碰撞时隙的概率^[13]为:

$$P_{\text{coll-succ}} = 1 - P_{\text{coll-fail}} = 1 - \left(\frac{1}{2}\right)^{(r-1)k} \quad (10)$$

于是 FCBPBS 机制预测成功的概率:

$$P_{\text{succ}} = \begin{cases} 1, & 0 \leq r \leq 1 \\ 1 - \left(\frac{1}{2}\right)^{(r-1)k}, & r \geq 2 \end{cases} \quad (11)$$

FCBPBS 机制预测失败的概率:

$$P_{\text{fail}} = \begin{cases} 0, & 0 \leq r \leq 1 \\ \left(\frac{1}{2}\right)^{(r-1)k}, & r \geq 2 \end{cases} \quad (12)$$

由式(11)可知,在发送预测后缀的标签数 r 一定的情况下,预测后缀的位数 k 越大预测成功的概率越大,但同时每个时隙的传输开销也会越大^[13]。

图 3 为 EPC Class-1 Generation-2 协议中 SA 防碰撞算法在 FCBPBS 机制下的标签识别过程。盘存周期调整

方式采用第 3 节的差分门限调整的方式,图中实例 $Q = 2$,标签数为 5。在时隙数为 1 时,假设这 5 个标签 SC 初始值设定为 1、1、0、2、5。与图 1BS 机制和图 2BPBS 机制比较,FCBPBS 机制识别完这 5 个标签所用时隙数大幅减少。

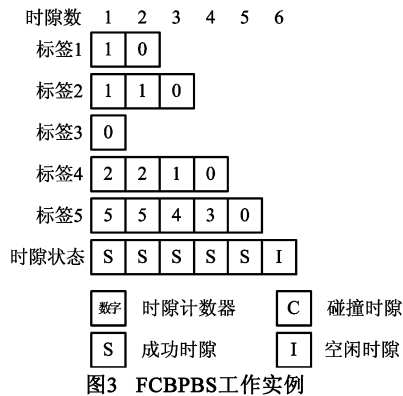


图3 FCBPBS工作实例

5 FBBS 算法设计与防碰撞实现

FBBS 防碰撞算法是在 Q 值差分门限调整机制和时隙前向检测后向预测二进制散列机制 FCBPBS 的基础上提出来的。FBBS 算法流程如图 4 所示。具体算法步骤如下:

①读写器向所有标签发送 Query(Q)指令。

②收到 Query(Q)指令或者 QueryAdjust(Q)指令且未被识别的标签进行初始化,在区间 $[0, 2^Q - 1]$ 中随机选择一个整数作为自身时隙计数器 SC 的值。

③未被识别的标签加载 SC 值到时隙计数器中,SC 为 0 和 1 的标签按照式(6)的格式返回数据。

④读写器检测碰撞检测位。若检测不到,则 $N_{\text{idle}} +$ 。若 $|N_{\text{coll}} - N_{\text{idle}}| = D(Q)$, Q 值减 1,转到⑦;否则读写器发送 QueryRep 命令,所有标签 SC 减 1,转到③。若能检测到碰撞检测位,转到⑤。

⑤读写器根据 $[RN16]_{SC=0}^k$ 判定当前时隙内响应的标签数。若只有一个标签,则当前时隙可读,读取响应的标签并使其静默并转到⑥。若有多个标签,当前时隙发生碰撞, $N_{\text{coll}} +$ 。若 $|N_{\text{coll}} - N_{\text{idle}}| = D(Q)$, Q 值加 1,转到⑦;否则按式(4)进行二进制散列处理,转到③。

⑥读写器根据 $[RN16]_{SC=1}^k$ 判定下一时隙内响应的标签数。若判断为没有标签响应,则 $N_{\text{idle}} +$ 。若 $|N_{\text{coll}} - N_{\text{idle}}| = D(Q)$, Q 值减 1,转到⑦;否则读写器发送 QueryRep 命令,所有标签 SC 减 2,转到③。若判断为只有一个标签响应,读写器判定下一个时隙有效,发送 QueryRep 命令,所有标签 SC 减 1,转到③。若判断为多个标签响应,则 $N_{\text{coll}} +$ 。若 $|N_{\text{coll}} - N_{\text{idle}}| = D(Q)$, Q 值加 1,转到⑦;否则按式(7)进行二进制散列处理,

转到③。
⑦如果 $Q = 0$ 或 $Q > 15$, 标签盘存结束; 如果 $0 < Q \leq 15$, 读写器则发送 $\text{QueryAdjust}(Q)$, 将 N_{idle} 、 N_{coll} 置 0, 转到②。

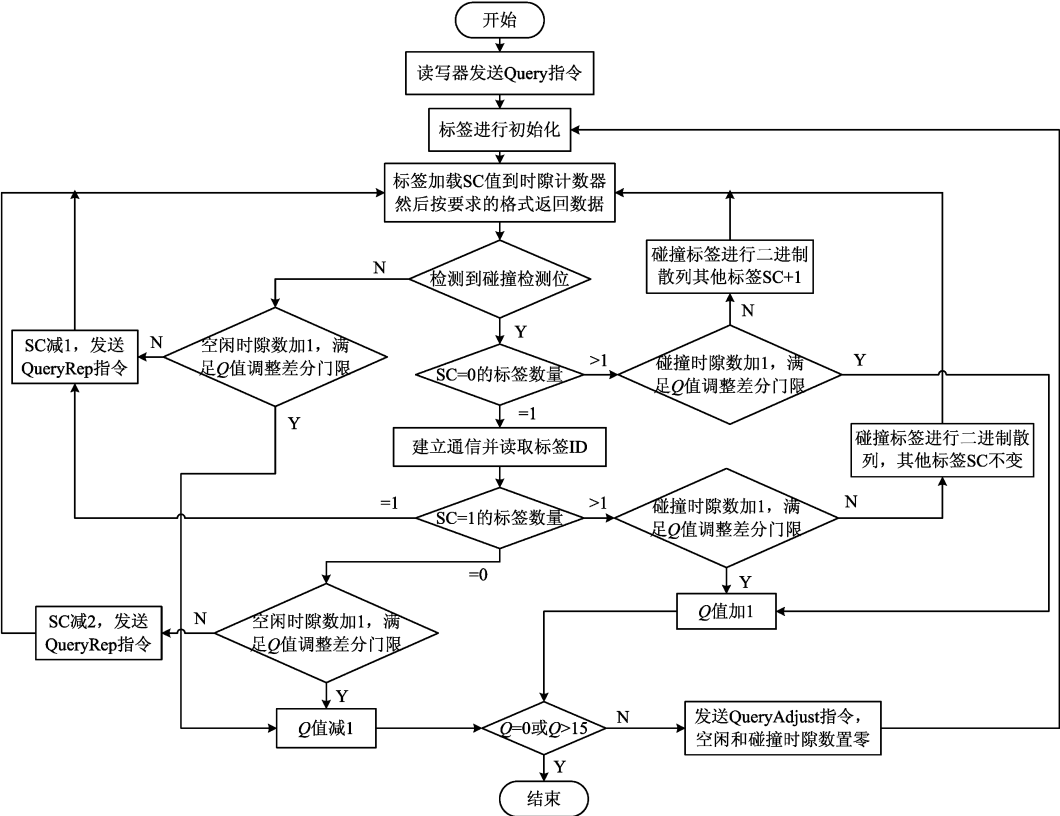


图4 FBBS算法流程图

6 FBBS 算法仿真实验

本文进行了如下 3 组仿真实验, 验证 FBBS 算法相比其他防碰撞算法的优越之处. 为确保实验数据的有效性, 每组实验数据均取 100 次实验的算术平均值.

①盘存周期调整方式仿真实验. 标签数以步长 100 从 100 到 1000 递增变化. 吞吐率定义为已被识别的标签数与识别完这些标签所用总时隙数的比值. 分别采用差分门限调整机制和经验 C 机制 ($C = 0.8/Q$) 对盘存周期进行调整, 比较在这两种调整机制下的系统吞吐率. 由如图 5 所示可知, 差分门限调整机制与经验 C

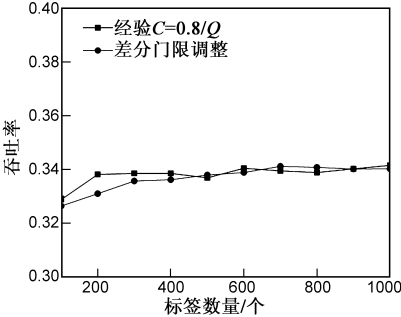


图5 差分门限调整机制下与经验C机制下的系统吞吐率比较

调整机制在系统吞吐率方面相差不大.

综上所述, 相同条件下差分门限调整机制对盘存周期的调整更为简单灵便, 而且避免了读写器处理完每个时隙后进行浮点运算, 提高了读写器的反应速度.

②盘存周期内时隙散列方式仿真实验. 标签数同样以步长 100 从 100 到 1000 递增变化, BS、BPBS 和 FCBPBS 这三种散列机制下空闲时隙数和碰撞时隙数与标签数之间的关系如图 6 和 7 所示.

当标签数量 N 为 1000 时, BS、BPBS ($k=1$)、BPBS ($k=2$)、BPBS ($k=3$)、BPBS (ideal) 和 FCBPBS ($k=3$) 机制产生的空闲时隙数如表 2.

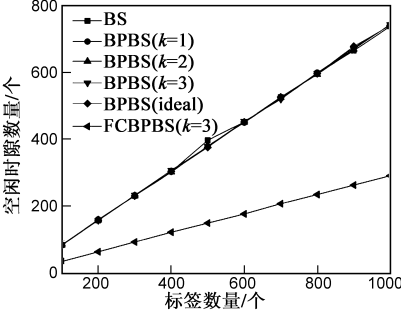


图6 三种散列机制下空闲时隙数量比较

如表 2 和图 6 所示,BS 和 BPBS 两种调整机制产生的空闲时隙数量几乎相同,而 FCBPBS($k=3$)机制大幅减少了空闲时隙;在 BPBS 机制下 $k=1, k=2, k=3$ 和后向预测的理想情况 BPBS(ideal)下产生的空闲时隙数量基本相同.这说明 BPBS 机制产生的空闲时隙数量与 k 无关,通过前向检测可大幅减少空闲时隙.

表 2 三种散列机制下空闲时隙数量比较($N=1000$)

BS	BPBS($k=1$)	BPBS($k=2$)
737	740	743
BPBS($k=3$)	BPBS(ideal)	FCBPBS($k=3$)
742	740	291

当标签数量 N 为 1000 时,BS、BPBS($k=1$)、BPBS($k=2$)、BPBS($k=3$)、BPBS(ideal)和 FCBPBS($k=3$)机制产生的碰撞时隙数如表 3.

如表 3 和图 7 所示,前五种调整机制产生的碰撞时隙数量依次递减;FCBPBS($k=3$)与 BPBS($k=1$)机制产生的碰撞时隙数量接近.这说明通过后向预测增加预测位数 k 可以减少碰撞的发生,而通过前向检测并不能有效减少碰撞的发生,反而由于前向检测对当前空闲时隙加速跳过,对于下一时隙的标签情况无法预测,导致碰撞时隙有所增加,从表中 FCBPBS($k=3$)调整机制下产生的碰撞时隙数量大于 BPBS($k=3$)可以得知.

表 3 三种散列机制下碰撞时隙数量比较($N=1000$)

BS	BPBS($k=1$)	BPBS($k=2$)
701	548	454
BPBS($k=3$)	BPBS(ideal)	FCBPBS($k=3$)
388	230	563

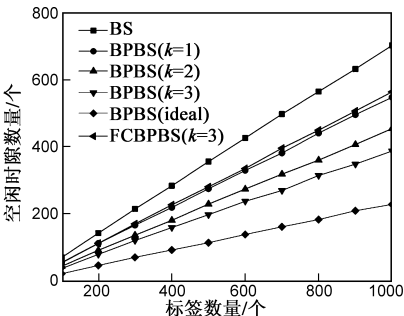


图7 三种散列机制下碰撞时隙数量比较

综上所述,通过前向检测可以有效地减少空闲时隙,后向预测可以减少碰撞时隙,后向预测的效果与预测位数有关,预测位数越多,预测效果越好. FCBPBS($k=3$)机制通过前向检测和后向预测相比 BPBS($k=3$)可以使无效时隙的总数减少到一个较低水平.虽然 FCBPBS($k=3$)机制由于进行前向检测和后向预测使标

签单次的传输开销大于 BPBS 机制和 BS 机制,但通过前向检测和后向预测大幅减少了无效时隙,减小了标签因发生碰撞而重传的概率,所以此时标签平均单次传输开销低于 BPBS 机制和 BS 机制.

③识别时延与吞吐率仿真实验.标签数量以步长 100 从 100 到 1000 递增变化,将 FBBS($k=3$)算法的识别时延和系统吞吐率与 FSA(256)、DFSA、SA 等几种经典的 ALOHA 算法进行比较.

当标签数量 N 为 1000 时,这几种算法所需的时隙数如表 4 所示.由表 4 和图 8,可以看出 FSA 算法所需要的时隙数量最多,DFSA 次之,随着标签数量的增多 FSA 和 DFSA 所需要的时隙数迅速增加,而 FBBS($k=3$)算法所用的时隙数量最少.

表 4 FSA、DFSA、SA、FBBS 算法时隙数比较($N=1000$)

FSA(128)	FSA(256)	DFSA	SA	FBBS($k=3$)
44643	5593	5365	2931	1854

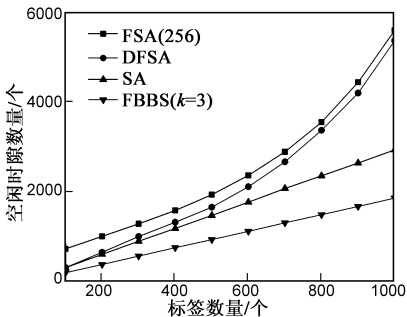


图8 FBBS($k=3$)与其他几种算法的识别时延比较

图 9 显示了 FSA(256)、DFSA、SA、FBBS($k=3$)4 种算法在不同标签数量下的吞吐率. FBBS($k=3$)算法的系统吞吐率最高,可达到 52% 以上, FSA(256)算法的系统吞吐率最低,而 DFSA 算法通过动态调整帧长,使得其系统吞吐率略高于 FSA(256);SA 算法的系统吞吐率仅次于 FBBS($k=3$),稳定在 33% 左右.

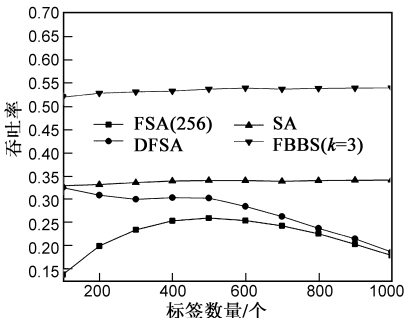


图9 FBBS($k=3$)与其他几种算法的吞吐率比较

综上所述,FBBS($k=3$)算法通过前向检测、后向预测和二进制散列机制可以大幅减少空闲时隙和碰撞时隙的发生,其识别时延最小、系统吞吐率最高.

7 结束语

在研究了 EPC Class-1 Generation-2 协议中 SA 防碰撞算法的基础上,提出了基于 Q 值差分门限调整和时隙前向检测后向预测二进制散列的 ALOHA 算法(FBBS 算法).该算法简化了协议中盘存周期参数 Q 值的调整方案,采用二进制散列的方法实时解决碰撞,通过在标签返回数据中设置前向碰撞检测位和预测后缀,有效地减少无效时隙的出现.仿真实验结果显示,提出的 FBBS 算法较大幅度减少了无效时隙占用信道的时间,系统吞吐率从常规的 33% 稳定提高到 52% 以上,适合在标签数量较多、实时性要求较高的物联网 RFID 终端应用领域.

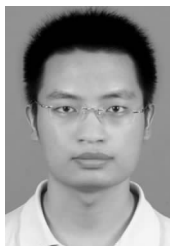
参考文献

- [1] Juels A. RFID security and privacy: a research survey[J]. IEEE Journal on Selected Areas in Communications, 2006, 24(2): 381 – 394.
- [2] 宁焕生,张瑜,刘芳丽,等.中国物联网信息服务系统研究[J].电子学报,2006,34(12A):2514 – 2517.
NING Huan-sheng, ZHANG Yu, LIU Fang-li, et al. Research on china internet of things' services and management[J]. Acta Electronica Sinica, 2006, 34(12A): 2514 – 2517. (in Chinese)
- [3] 钱志鸿,王义君.物联网技术与应用研究[J].电子学报,2012,40(5):1023 – 1029.
QIAN Zhi-hong, WANG Yi-jun. IOT technology and application[J]. Acta Electronica Sinica, 2012, 40(5): 1023 – 1029. (in Chinese)
- [4] Grover A, Berghel H. A survey of RFID deployment and security issues[J]. Journal of Information Processing Systems, 2011, 7(4): 561 – 580.
- [5] 王雪,钱志鸿,胡正超,等.基于二叉树的 RFID 防碰撞算法的研究[J].通信学报,2010,31(6):49 – 57.
WANG Xue, QIAN Zhi-hong, HU Zheng-chao, et al. Research on RFID anti-collision algorithm based on binary tree[J]. Journal on Communications, 2010, 31(6): 49 – 57. (in Chinese)
- [6] 张学军,蔡文琦,王锁萍.改进型自适应多叉树防碰撞算法研究[J].电子学报,2012,40(1):193 – 198.
ZHANG Xue-jun, CAI Wen-qi, WANG Suo-ping. One anti-collision algorithm based on improved adaptive multi-tree search[J]. Acta Electronica Sinica, 2012, 40(1): 193 – 198. (in Chinese)
- [7] Wu H, Zeng Y. Bayesian tag estimate and optimal frame length for anti-collision aloha RFID system[J]. IEEE Transactions on Automation Science and Engineering, 2010, 7(4): 963 – 969.
- [8] Lei Z, Yum T S P. Optimal framed aloha based anti-collision algorithms for RFID systems[J]. IEEE Transactions on Communications, 2010, 58(12): 3583 – 3592.
- [9] 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.
- [10] Schoute F. Dynamic frame length aloha[J]. IEEE Transactions on Communications, 1983, 31(4): 565 – 568.
- [11] 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.
- [12] Thomas F L P, Gaia M, Chiara P. Anticollision protocols for single-reader RFID systems: temporal analysis and optimization[J]. IEEE Transactions on Mobile Computing, 2011, 10(2): 267 – 279.
- [13] 李萌,钱志鸿,张旭,等.基于时隙预测的 RFID 防碰撞 ALOHA 算法[J].通信学报,2011,32(12):43 – 50.
LI Meng, QIAN Zhi-hong, ZHANG Xu, et al. Slot-predicting based aloha algorithm for RFID anti-collision[J]. Journal on Communications, 2011, 32(12): 43 – 50. (in Chinese)

作者简介



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、硕士生导师.研究方向: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



张留洋 男,1988 年 1 月出生,河南禹州人.现为江西理工大学信息工程学院硕士研究生.研究方向:射频传感系统理论、RFID 防碰撞算法.
E-mail: zhangliuyang2011@126.com