基于二分查找的动态帧时隙标签防冲突算法*

郭志涛[a], 李玮玮[b], 梁志刚[b], 顾军华[b]
(河北工业大学 a. 信息工程学院; b. 计算机科学与软件学院, 天津 300401)

摘 要: 在动态帧时隙算法中,根据标签到达基本符合泊松分布的特点,同时在分析其斜率特点的基础上,结合二分查找的思想,提出一种新颖的时隙调整算法,即基于二分查找的动态帧时隙标签防冲突算法,快速实现时隙调整。仿真实验表明,该方法能够显著减少识别次数,提高单位时间的识别率。

关键词: 二分查找识别; 射频识别; 动态帧时隙算法; 防冲突
中图分类号: TP393;TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2012)11-4287-03
doi:10.3969/j.issn.1001-3695.2012.11.073

Dynamic framed slotted tag anti-collision algorithm based on binary search

GUO Zhi-tao[a], LI Wei-wei[b], LIANG Zhi-gang[b], GU Jun-hua[b]
(*a. School of Information Engineering, b. School of Computer & Software, Hebei University of Technology, Tianjin* 300401, *China*)

Abstract: Among the dynamic frame slotted algorithm, according to the characteristic of tags arriving in line with the Poisson distribution, this paper analyzed slope apprentice, combined with the idea of binary search, and then proposed a new slot adjust algorithm: dynamic frame slot tag anti-collision algorithm based on binary search: to achieve quick slot adjustment. Simulation results show that the method can significantly reduce the number of recognition and improve the recognition rate.

Key words: binary search recognition; radio frequency identification; dynamic framed slotted algorithm(DFSA); anti-collision

ALOHA 算法是一种时分多路的存取方式,它将传输时间划分为若干时隙,实现了交互计算机传输。标签随机选择时隙进行数据传输,以突发脉冲序列方式接收和发送信号。

传统的动态帧时隙算法由最初的纯 ALOHA(P-ALOHA)演变而来,经历了时隙 ALOHA(S-ALOHA)算法,使识别率提高了一倍,最终发展为动态帧时隙 ALOHA(DFSA)算法,识别率有了进一步的提高。虽然 ALOHA 算法进行了一系列的改进和提高,但是标签的冲突问题依然是 RFID 系统中影响识别率的最主要问题,因此,如何更好地降低标签的冲突率依然是目前亟待解决的问题。

泊松分布在以往帧时隙调整策略中主要的应用体现在标签数的估计中,目的是为了应用 Schout 估计法来预测未识别标签数,但是并没考虑泊松分布的斜率特点应用于时隙调整中,选择 Q 值估算方法[1]也只是着重应用于对时隙的分配上。由于泊松分布的一个显著特点是在一个时间段内标签数量可能会过快地增加或减少,传统帧时隙 ALOHA 方法忽略了这一特点,所以存在调整方法灵敏度不高、调整速度慢的特点;另外,标签随机选择时隙传输也在一定程度上增加了冲突的可能性,成为影响识别效率的一个因素。本文针对这两种情况,通过分析泊松分布的特点提出了基于二分查找的动态帧时隙标签防冲突算法。

1 动态帧时隙标签防冲突算法及存在问题分析

1.1 传统 ALOHA 防冲突算法

最初的 P-ALOHA 算法是最简单的也是最容易操作的方法。该方法中标签成功传输的条件是前后连续的两个时隙周期内没有信道占用。标签传输可能发生成功传输、部分冲突、完全冲突三种状况。由于成功传输的情况大大牺牲了信道的利用率,从而使吞吐量(归一化)的最大值大仅为 0.18。

S-ALOHA 算法改进了 P-ALOHA 算法的这一缺点,采用一种广播同步脉冲序列的方式进行标签数据传输,要求每个 tag 只能在时隙的开始时刻进行数据通信,避免了部分冲突的情况,这使得冲突周期缩短了一个时隙周期,归一化的吞吐量 ρ 和总负载 G 的关系由 $\rho = e^{2G}$ (P-ALOHA)改变为 $\rho = Ge^{-G}$,比纯 ALOHA 提高了 1 倍,其最大值大约为 0.37。P-ALOHA 和 S-ALOHA 吞吐量对比关系如图 1 所示。

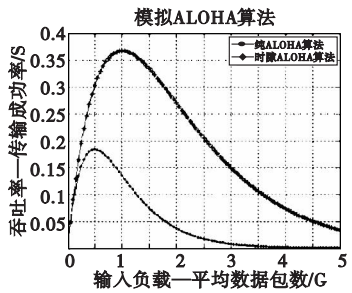


图1 *P*-ALOHA和*S*-ALOHA吞吐量对比关系

动态帧时隙 ALOHA 算法是对前两种方法的进一步改进。该算法把一个时间段划分成若干个帧,每个帧再划分成若干个

收稿日期: 2012-03-19; **修回日期**: 2012-04-26 **基金项目**: 天津市应用基础及前沿技术研究计划基金资助项目(11JCYBJC0020)

作者简介: 郭志涛(1979-),男,河北唐山人,博士,主要研究方向为视频识别、智能算法;李玮玮(1984-),女,河北邯郸人,硕士,主要研究方向为智能信息处理(bddylww@126.com);梁志刚(1982-),男,博士,主要研究方向为智能信息处理;顾军华(1966-),男,河北石家庄人,教授,主要研究方向为智能信息处理、宽带网络应用技术、虚拟现实理论与方法.

时隙,待识别的标签在每一帧内随机选择时隙进行数据传输。所谓的动态是指通过冲突时隙统计来估计标签数目,预测帧长和时隙数,自动地对时隙进行调整。为了减少时隙内的冲突,有的采用了构造 hash 函数对标签进行时隙分配[2]。

1.2 典型 DFSA 算法中存在的问题分析

在经典的 DFSA 算法中,最关键的方法是利用 Schout 方法对标签数的预测,从而调整下一帧长。在帧内时隙的调整中,统计冲突时隙数、空时隙数及成功传输时隙数。当信道(或时隙)没有标签进行传输时,时隙数减 1;当时隙数为 0 时,识别过程结束。通过类似的调整算法可以看出,其帧内调整策略为:a)无标签发送,时隙数减 1;b)产生冲突,时隙数加 1。这种方法比较精确地计算了冲突情况,在帧内时隙的调整中却存在相应的弊端,即:a)灵敏度不高;b)调整速度慢。下面本文通过分析泊松分布与标签分布的特点,从而提出这两种弊端的解决方法。

2 二分查找 DFSA 算法

2.1 泊松分布与标签分布特点分析

标签到达的概率与泊松分布曲线存在一致性,因此可以通过分析泊松分布曲线来分析标签的分布,从而指导时隙调整方法。对于泊松分布,在数学领域是这样描述的:如果稀有事件 A 在每个单元(设想为 n 次实验)内平均出现 λ 次,那么在一个单元(n 次)的实验中,稀有事件 A 出现次数 x 的概率分布服从 Poisson 分布。可以理解为在 T 时刻内共计 n 个标签,其中标签出现次数为 x 的概率服从泊松分布。泊松分布的曲线如图 2 所示,图中横轴表示出现次数 x,纵轴表示概率。由图 2 可知,泊松分布最显著的特征是:上升阶段和下降阶段的增幅较大,也就是说其斜率的绝对值较大,即单位时间内,标签数量增加或减少的速度比较快。根据泊松分布的这个特点,要更准确地估计标签数,就需要根据泊松分布的曲线特点对其斜率进行模拟,这样才能更有效地预测下一次标签识别需要的时隙数,减少标签传输过程中的冲突问题,以提高识别率。

通过对泊松分布特点的分析,要提高识别率,减少标签冲突,就需要找到合理的方法来拟合泊松分布的斜率。如图 3 所示,对泊松分布的走势图进行进一步的斜率分析。对泊松分布曲线某一点做切线,斜率的计算是由 $\Delta y/\Delta x$ 得出。但是要精确地计算泊松分布的斜率是比较复杂的,计算的时间复杂度比较大,所以采用模拟近似的方法,即采用二分查找方法两边取中的策略来近似模拟其斜率。本文将二分查找方法应用于传统动态帧时隙标签防冲突方法中,提出了二分查找 DFSA 算法,用二分查找的思想估计时隙数,以求降低标签冲突率,提高标签识别率。该算法适合于标签数量较少的场合,对于标签数量较大的情况,通常采用分组的方法,也有的分析冲突因子与冲突时隙的关系,拟合其曲线来减少冲突[3]。

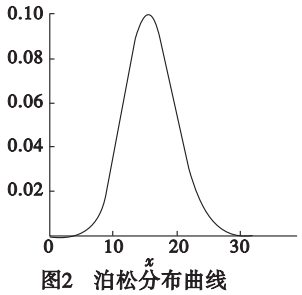


图2 泊松分布曲线

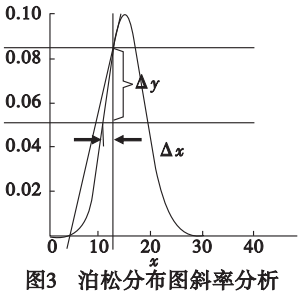


图3 泊松分布图斜率分析

2.2 标签估计

根据现在应用得最广泛的 Schoute 估算方法[4],假设时隙数为 F,标签总数为 N,则有 n 个标签选择同一个时隙的概率为[5]

$$P_{N,1/F}(n)=C_N^n\left(\frac{1}{F}\right)^n\left(1-\frac{1}{F}\right)^{N-n}=C_N^n\frac{(F-1)^{N-n}}{F^N} \quad (1)$$

当某一帧识别结束后,取 $n=0$,$n=1$ 则可统计出空时隙数目 S_0、成功识别时隙数 S_1 的概率,如式(2)(3)所示。

$$P_{N,1/F}(0)=C_N^0\left(\frac{1}{F}\right)^0\left(1-\frac{1}{F}\right)^N=C_N^0\frac{(F-1)^N}{F^N}=\left(\frac{F-1}{F}\right)^N \quad (2)$$

$$P_{N,1/F}(1)=C_N^1\left(\frac{1}{F}\right)\left(1-\frac{1}{F}\right)^{N-1}=$$

$$C_N^1\frac{(F-1)^{N-1}}{F^N}=\frac{N}{F}\left(\frac{F-1}{F}\right)^{N-1} \quad (3)$$

每个时隙的状态可以分为空时隙、成功识别时隙和冲突时隙三种情况。因此根据统计规律,冲突时隙数 S_c 的概率为

$$P_{Sc}(n)=1-P_{N,1/F}(0)-P_{n,1/S}(1)=$$

$$1-\left(1+\frac{N-1}{F}\right)\left(1-\frac{1}{F}\right)^{N-1}\quad n\geqslant 2 \quad (4)$$

由此可见,时隙数目与标签数目 N 之间存在比较确定的概率统计关系[6],这也是标签估计的依据。

2.3 二分查找 DFSA 算法描述

本文中用到的参数符号有吞吐率 g、冲突时隙数 S_c、成功传输时隙数 S_1、空闲时隙数 S_0、当前时隙 F_{slot}、最少时隙数 F_{min}、最大时隙数 F_{max}。

2.3.1 标签设计

传统 DFSA 方法中,待识别的标签在每一帧内随机选择时隙进行数据传输,高度的随机性,使同一时隙被不同标签选择的概率大大提高。为避免这一问题,同时减少阅读器负担,降低系统功耗,节约时间,本文对标签进行了简单的设计。每个标签数据分为唯一的 ID 识别码、数据信息、时隙号、传输标志四部分,如图 4 所示。其中:S_n 表示标签选择传输的时隙号,用于记录分配给标签的时隙;S 表示标签是否发生冲突(0 表示未发生冲突,传输成功;1 表示冲突,传输不成功)。发生冲突的标签(即标签状态 $S=1$)将在下一帧中被重新分配时隙,同时标签状态设置为 $S=0$。

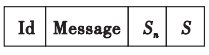


图4 标签设计

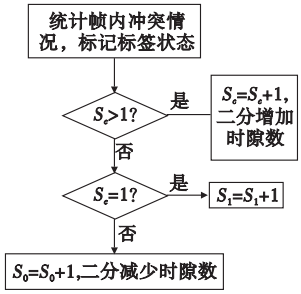


图6 二分调整时隙流程

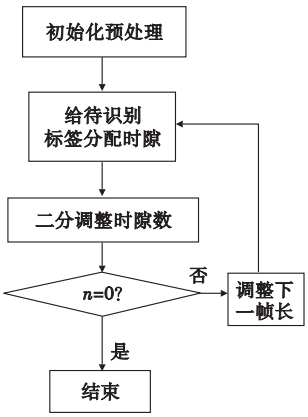


图5 二分查找算法流程

2.3.2 主要算法过程描述

算法基本过程如下:

a)初始化,设置初始标签数(如 100)和时隙数(F_{min}、F_{max}、F_{slot})。

b)给标签分配时隙。

c)记录冲突情况。标记冲突标签状态 S_0 包括统计冲突时隙数 S_c,成功识别时隙数(成功识别标签数)S_1,空时隙数 S_0。

d)判断冲突时隙数是否为 0,相应二分调整帧内时隙数,增加或减少时隙数。

e)当前帧识别结束,根据上一帧内识别标签的冲突情况估计并调整下一帧长。

f)判断标签是否识别结束($n=0$)。未结束则执行 b)。

具体算法流程如图 5、6 所示。其中二分调整时隙数流程如图 6 所示。

2.3.3　二分查找 DFSA 时隙调整

二分查找 DFSA 时隙调整是整个算法中的核心部分,该部分主要完成两个功能:a)统计冲突的时隙数、标签成功传输时隙数;b)用类似二分查找方法调整时隙数。具体流程如图 6 所示。其中,帧内时隙数调整分为二分增加时隙数和二分减少时隙数。初始化最大时隙数 $F_{\max}$(如 32)与最小时隙 $F_{\min}$(如 1)。二分增加时隙数计算是对当前时隙数与最大时隙数的折半,如式(5)所示:

$$F_{\text{slot}} = \text{fix}((F_{\text{slot}} + F_{\max})/2 \tag{5}$$

二分减少时隙数计算是对当前时隙数与最小时隙数的折半,如式(6)所示:

$$F_{\text{slot}} = \text{fix}((F_{\text{slot}} + F_{\min})/2) \tag{6}$$

在调整下一帧长时,通过冲突情况计算出下一帧的最大时隙,根据 Schout 估算方法计算最大时隙数,估计第 $i(F_{\min} \leqslant i \leqslant F_{\max})$时隙系统中未被读写器识别的标签数目期望值,如式(7)所示:

$$\hat{N} = 2.3922 S_c \tag{7}$$

确定下一帧的最大帧长为冲突时隙数与未识别标签期望值的乘积,如式(8)所示:

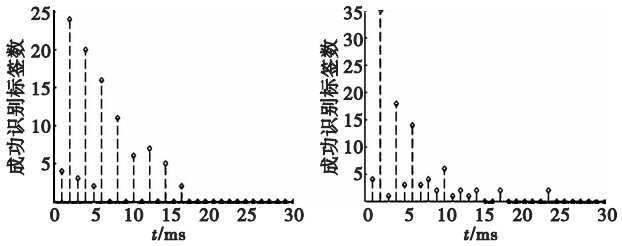
$$F_{\max} = S_c \times \hat{N} \tag{8}$$

其中:S_c 为根据识别情况统计的碰撞时隙数。然后初始化下一帧时隙数,如式(9)所示:

$$F_{\text{slot}} = \text{fix}((F_{\min} + F_{\max})/2) \tag{9}$$

3　算法仿真结果分析

本文将二分查找识别 ALOHA 算法与 DFSA 算法进行仿真对比。本实验在 MATLAB 平台环境下进行模拟,仿真实验以初始标签数目 100 为例,最大时隙数 $F_{\max}=32$,最小时隙数 $F_{\min}=1$,实验运行 50 次,取平均值,分别测试两种算法的识别时间及冲突率。图 7、8 分别是对二分查找识别算法和 DFSA 算法的标签识别过程和冲突时隙数的仿真。通过对比发现:在单位时间内,二分查找 DFSA 算法相对于 DFSA 算法的平均识别标签数增多,冲突时隙数大约减少 36.4%,平均识别时间比 DFSA 算法减少 5 ms。二分查找识别算法在标签识别前期($0<t<4$ ms)的冲突率与 DFSA 算法比较接近,随标签数量的增多,DFSA 识别算法的冲突时隙数比二分查找 DFSA 算法减少得慢,如表 1 所示。在识别后期待识别标签数逐渐减少,DFSA 算法冲突时隙数的图形趋于平缓,没有明显变化,这说明二分查找 DFSA 算法在后期对标签的识别效率高于 DFSA 算法。可见二分查找识别算法的时隙调整速度比 DFSA 算法快,时隙数调整的敏感性优于 DFSA 算法。



(a)二分查找识别算法　　(b)DFSA算法

图7　成功识别标签数与时间关系对比

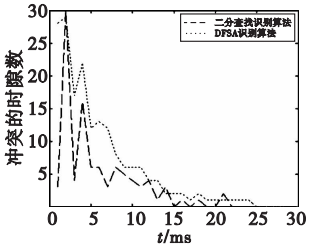


图8　冲突时隙数对比

表 1　二分查找 DFSA 与 DFSA 算法对比

时间 t/ms	成功识别标签数		冲突时隙数	
	二分查找 DFSA	DFSA	二分查找 DFSA	DFSA
1	4	4	4	28
2	24	35	—	—
3	3	1	30	29
4	20	18	5	17
5	3	3	6	13
6	16	14	—	—
7	0	3	3	10
8	12	4	5	6
9	0	2	—	—
10	7	6	4	5
11	0	1	—	—
12	7	2	4	4
13	0	1	2	4
14	6	2	3	2
15	0	0	0	2
16	2	0	1	2
17	—	2	0	1
18	—	—	1	2
19	—	—	0	1

4　结束语

基于二分查找的 DFSA 算法充分利用泊松分布的斜率特点,并通过二分查找方法提高识别速率,比较适合于标签数量较少且不需要对标签进行分组的场合。通过 MATLAB 模拟与动态帧时隙 ALOHA(DFSA)算法进行对比,二分查找 DFSA 算法降低了识别时间,减少了冲突次数,使识别效率有所提高。在实际应用中如何设置标签识别的平均时间,还需要进一步的研究。

参考文献:

[1] 颜元,武岳山,熊立志. 一种新型多标签估算方法[J]. 计算机应用研究,2012,29(3):930-932.

[2] 郭志涛,程林林,周艳聪,等. 动态帧时隙 ALOHA 算法的改进[J]. 计算机应用研究,2012,29(3):907-909.

[3] 贺洪江,丁晓叶,翟耀绪. 一种基于碰撞因子的 RFID 标签估算方法[J]. 计算机应用研究,2011,28(11):4131-4133.

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

[5] 张玉平,赵东东,洪辉. UHF 频段 RFID 系统防碰撞算法研究[J]. 微计算机信息,2009,25(3-2):228-229,237.

[6] 邓晓,何怡刚,向阳,等. 一种新的 RFID 标签数目估算方法[J]. 计算机工程与应用,2008,44(31):142-144.