RFID 系统中一种动态帧时隙算法的研究与仿真①

Simulation and Research on an RFID Dynamic Framed Slotted Algorithm

杜俊宇 何 宁 (桂林电子科技大学 信息与通信学院 广西 桂林 541004)

摘 要: 在多个电子标签的识别过程中，利用防碰撞算法解决标签的碰撞问题是射频识别系统中的一项关键技术。对一种动态帧时隙 Aloha 算法进行了描述与仿真，根据估算所得的标签数,结合标签的 EPC 对标签进行了分组，将分组码作为激活码每次激活一组标签而屏蔽其它标签，对激活的这组标签用 FSA 算法进行识别。采用这种方法可将每次响应的标签数目限制在一定范围之内，并且与最大帧长基本匹配，从而在各个识别周期之内不需要再调整帧长，简化了算法且保证了较高的识别率。仿真结果表明：在标签数目较大时，阅读器的识别效率可以稳定在 30.5%—36.8%之间，从整体上提高了系统的效率和可靠性。

关键词: 射频识别 防碰撞 动态帧时隙算法 标签分组

1 引言

射频识别(Radio Frequency Identification，RFID)技术是当今非接触的自动识别技术中发展非常迅速的一种技术。它利用射频方式进行非接触双向通信，以达到识别目的并交换数据，大大提高了定位、管理等应用的处理速度。然而，在射频识别系统工作时，经常有较多的应答器同时处于阅读器的作用范围内。这些应答器的数据同时传送到阅读器时，导致各应答器之间传输的信号产生相互的干扰，读写器接收不到正确的信息，这就是所谓的碰撞问题。因此，如何较好的解决碰撞问题成为了 RFID 系统的关键技术之一。

目前，在基于 Aloha 的算法中，动态帧时隙算法(Dynamic Framed Slotted Aloha,DFSA)由于操作简单和性能良好而成为目前最常用的算法之一。通常，当标签数量增多时，增大帧长可在一定程度上改善系统性能。但实际应用中帧长并不能无限增加，因此，DFSA 算法也存在所需时隙数增长过快的问题。在已有的 DFSA 改进算法中，分组问题大都是基于概率的分组思想，在对标签进行分组的方法上不够明确[1]。

针对目前 DFSA 算法中分组数调整方法的不明确性，本文借鉴了二进制树中的分组思想，在分组过程中与标签的 EPC 关联起来，可以对标签进行较为均匀的分组。在标签数量很大时，通过明确的分组，有效的限制每次响应的标签数量，使每次响应的标签数都与帧时隙算法的帧长相匹配，从而获得较高的标签识别效率。

2 ALOHA 算法

2.1 时隙 ALOHA 算法

时隙 ALOHA(Slotted ALOHA)算法是 ALOHA 算法的改进算法。这种算法在 ALOHA 算法的基础上把时间分成多个离散时隙，并且每个时隙长度要大于标签回复的数据长度，标签只能在每个时隙内发送数据。在时隙 ALOHA 算法中，标签或成功发送或完全碰撞，避免了纯 ALOHA 算法中的部分碰撞，提高了信道的利用率。

2.2 帧时隙 ALOHA 算法[2]

虽然时隙 ALOHA 算法提高了系统的吞吐率，但是当标签大量地进入系统时，该算法的效率并不高，因此帧时隙(Framed Slotted ALOHA,FSA)算法被提出。FSA 算法是在时隙 ALOHA 算法的基础上把 N 个时隙组成一帧，标签在每个帧内随机选择一个时隙发送数据。读写器判断标签是否被识别，发生碰撞的标

① 收稿时间:2009-03-10

签进入下一识别周期，直到所有标签被识别。FSA 算法存在一个缺点，即当标签数量远大于时隙数时，读取标签的时间会大大增加，而当标签数远小于时隙数时，会造成时隙的浪费。

2.3 动态帧时隙 ALOHA 算法

动态帧时隙算法(DFSA)弥补了 FSA 算法的不足，它根据每帧中的空闲和碰撞情况动态调整帧长以提高识别效率。目前改进的 DFSA 算法大部分是根据标签碰撞和空闲的概率调整帧长[6]。例如，当识别周期内碰撞概率大于 0.7 时帧长增加一倍，当空闲概率大于 0.3 时就将帧长减半。碰撞的随机性使采用该方法时系统稳定性变差，而且当标签数量增多时，虽然增大帧长可在一定程度上改善系统性能，但实际应用中帧长并不能无限增加，因此，DFSA 算法也存在所需时隙数增长过快的问题[5]。

3 改进的动态帧时隙算法

为了弥补 DFSA 算法的不足，可以将阅读器范围内的标签进行有效分组，通过限制每次响应的标签数来实现[6]。本文提出的算法的主要思想是：在有大量标签出现在阅读器范围的时候，首先估算总的标签数，再结合标签的 EPC 对标签进行分组，将分组码作为激活码每次激活一组标签而屏蔽其它标签，从而在每个识别周期内有效地限制了响应的标签数，再用 FSA 算法对激活的标签进行识别。标签的估算方法如下：

首先假设帧长和读写器范围内的标签数为 N 和 n，假设分别存在 E、S、C 个时隙(E+S+C=N),其中，E 表示这 E 个时隙都空闲没有标签发送数据，S 表示在这 S 个时隙中分别只有一个标签成功发送数据，C 表示在这 C 个时隙中，每个时隙中都发生了碰撞，没有成功发送数据。可得如下公式[4]：

$$E=\binom{N}{1}\left(1-\frac{1}{N}\right)^{n}=N\left(1-\frac{1}{N}\right)^{n} \tag{1}$$

$$S=\binom{n}{1}\left(1-\frac{1}{N}\right)^{n-1}=n\left(1-\frac{1}{N}\right)^{n-1} \tag{2}$$

$$C=N-E-S \tag{3}$$

分别定义 P_e 为在一帧中时隙空闲的概率、P_s 为成功发送标签的概率、P_c 为发生碰撞的概率。由上述公式可得：

$$P_e=\left(1-\frac{1}{N}\right)^{n} \tag{4}$$

$$P_s=\frac{n}{N}\left(1-\frac{1}{N}\right)^{n-1} \tag{5}$$

$$P_c=\frac{C}{N}=1-\left(1-\frac{1}{N}\right)^{n}-\frac{n}{N}\left(1-\frac{1}{N}\right)^{n-1} \tag{6}$$

由于帧长一般为 2、4、8、16…256，我们在设定初始帧长时，可以根据实际情况，设定一个较合理的初始值，使其尽量接近标签数 n。在帧长 N 被设定之后，可以通过阅读器向范围之内的标签发送相关指令，根据反馈的相关数据初步得到 E 与 S 的值，再由公式(3)求出 C，从而得出 P_c，再由公式(6)估算出标签数 n。

由文献[3]可知，当帧长 N 与标签数 n 基本相等时，系统的吞吐率最大，识别效率最高。因此，在确定分组数时，我们将估算出的标签数 n 与设定的初始帧长 N 做比较，如果 n>N 则相应的增大帧长 N，当 n> N_{max} 时，则开始对标签进行分组。在对标签分组时，与标签的 EPC 结合起来确定标签的分组数。具体的分组方法如下：

一个标签的编码中，除去产品信息等相同的编码位，剩余的是标签的有效识别位。例如，假设帧长 N==128 时，经过程序估算在阅读器范围内有大约 500 个标签，标签的有效识别位为：1000000000—1111111111，我们可以将标签从高位开始分为 4 组：100xxxxxxx,101xxxxxxx,110xxxxxxx,111xxxxxxx，标签有效识别位的前三位(100、101、110、111)做为分组码，除去分组码外，有效识别位还剩 7 位，则可知每组中被激活的最大标签数 n' <=128，与帧长基本接近，因此在每一个识别周期内，可以保证较高的识别效率。

读写器在每个识别周期首先执行 Request 命令，将标签的分组码作为激活码向区域内的标签发送。有效识别位的前三位与激活码相同的标签被激活，其它的标签则未被激活，从而限制了响应的标签数，对第一组被激活的标签用 FSA 算法进行识别，识别完成之后依次激活第二组、第三组、第四组，直到所有标签被识别完毕。

当标签数较小时，采用改进的 DFSA 算法和常用的 DFSA 算法系统性能相差较小。但在标签数量很大

时，前者会使系统更加稳定。这主要是因为，采用这种方法分组明确，不仅有效的限制了响应的标签数，而且可以大概预知每组中的标签数，使每个识别周期内的标签数与帧长基本接近，使得每个分组的识别效率保持在一个较高的水平，系统的总体性能基本稳定。算法流程图如图 1 所示。

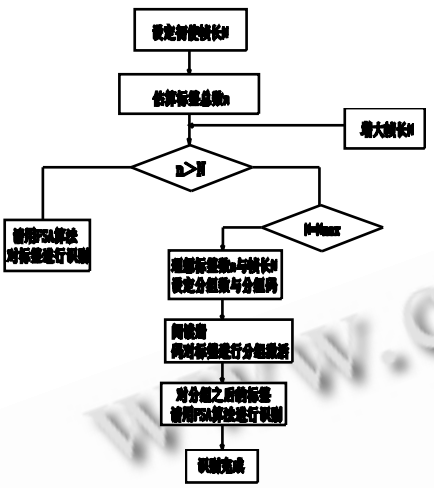

图 1　算法流程图

4　仿真

4.1 仿真过程

仿真过程主要包括理论计算和编程实现两部分，是在 MATLAB7.0 的环境下进行的。主要分为以下几步：

第一步，标签数估计:根据帧中成功发送的时隙数 S 和空闲时隙数 E，由公式(3)计算出碰撞时隙数 C，从而得出时隙的碰撞概率 P_c，由公式(6)构造函数 TagN(输入参数为当前帧长 N 和碰撞概率 P_c，输出为估计所得的标签数 n)。

第二步，构造函数 SlotALOHA，(输入为标签数 n、最大帧长 N_{max}、标签与读卡器交换数据时间 T0，输出为吞吐量 G)。

第三步，构造帧长函数 FrameLength，(参数值为标签数 n 和当前帧长 N)。将 n 与当前帧长 N 做比较，如果 N 小于 n 则调整帧长，并返回第一步继续进行估计。直到帧长 N 与标签数匹配，则调用函数 SlotAloha；如果当前帧长已调整至系统最大帧长，则执行分组函数 GROUP 对标签进行分组。

第四步，构造分组函数 GROUP(参数值为标签数 n 和最大帧长 N_{max})。

第五步，每组分别调用执行 SlotAloha 函数。

第六步，标签数目 n 从小到大改变时，分别执 SlotAloha 函数，保存与之对应的各组吞吐率 G，做出仿真曲线。

4.2 仿真结果分析

我们分别将 FSA 算法、DFSA 算法和本文改进的算法在 Matlab 环境下进行了仿真，并做了对比分析。图 2 中比较了这三种算法在标签数从 0 递增到 1000 时，全部标签识别完成需要消耗的时隙数。从图中可以看出，由于 FSA 算法不能动态调整帧长，在标签数目急剧增大时，消耗的时隙数几乎呈指数增长；DFSA 算法在标签数超过 600 时，随着标签数的继续增大，所消耗的时隙数增长也较快；而改进后的算法，所需时隙数与标签数基本呈一种斜率较小的线性关系。

图 3 为这三种算法的性能比较。从图中可以看出，FSA 算法随着标签数目的增大，标签识别效率急剧下降；DFSA 算法随着标签数目的增大，存在帧长增长过快的问题，由于帧长是根据碰撞和空闲的概率进行调整的，碰撞的随机性使采用该方法时系统性能较差；改进的算法对标签进行了较为合理的分组，使每组内的标签数目与帧长 N 基本匹配，从而保证在每个识别周期之内，标签的识别效率都基本保持在 30.5%—36.8%之间，这是采用 Aloha 算法能达到的一个较高的识别效率。

5　结语

文中算法与其他改进的动态帧时隙算法的区别主要有以下两点：

首先，分组方法明确，对标签分组时，没有采用取模的思想，而是采用了二进制树算法的思想，是根据标签的有效识别位进行分组的，是 ALOHA 算法与二进制算法的一个结合。

其次，分组是在帧长调整到最大之后才进行的，分组数是由估计所得的标签数并结合最大帧长来确定的。这使得分组之后每组激活的标签数目与帧长基本匹配，因此在每个识别周期之内不需要再进行帧长的调整，从而简化了算法，这是与其他改进型算法的一个主要区别。

由于分组数依赖于估计所得的标签数，因此，标签数目估计的是否准确，对分组数是否合理会产生严重影响。建立更加合理的标签数目估计模型，是本文算法今后需要改进的地方之一。

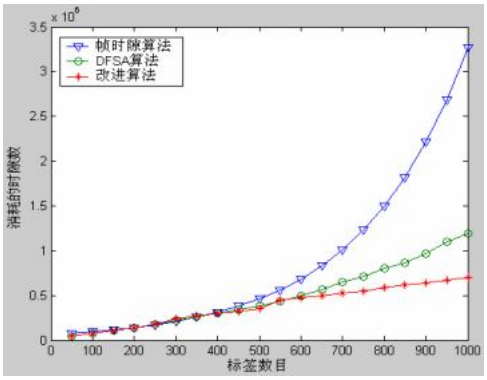

图 2　不同算法消耗的时隙数

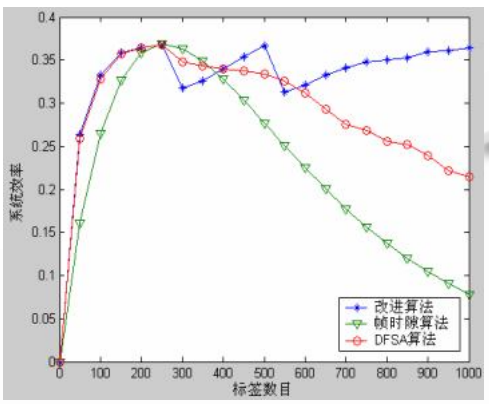

图 3　三种算法系统效率比较

参考文献

1 聂赛如,刘有源.RFID 防碰撞算法研究现状与发展趋势.技术与方法, 2008,27(3):65－68.

2 徐圆圆,曾隽芳,刘禹.基于 Aloha 算法的帧长数改进研究.计算机应用, 2008,28(3):588－590.

3 陆端,王刚,闫述.改进 ALOHA 算法在 RFID 多目标识别中的应用.微计算机信息, 2006,22(11-2):231－233.

4 陈祖爵,秦栋,王洪金.改进型帧时隙 ALOHA 防碰撞算法研究.无线通信技术, 2008,3:50－54.

5 Cha JR, Km JH. Novel anti-collision algorithms for fast object identification in RFID system. Proc. of the 11th International Conference on Parallel and Distributed Systems Workshops. Washington: IEEE Computer Society, 2005.63－67.

6 Hwang TW, Lee BG, Km YS. Improved anticollision scheme for high speed identification in RFID system.Proc. of the first international Conference on Innovative Computing, Information and Control. Washington: IEEE Computer Society, 2006. 135－139.