

跳跃式二进制树形搜索防碰撞算法及其分析

洪海亮, 金 杰, 矫亭亭

(1 天津大学 电子信息工程学院, 天津 300072; 2 南开大学 经济与社会发展研究院, 天津 300071)

摘 要: 在介绍 RFID 系统中防碰撞算法的基础上, 提出了跳跃式二进制树形搜索防碰撞算法. 通过设置 LIFO 栈, 充分利用碰撞节点信息使其能快速、有效地识别 RFID 标签. 通过对其分析, 该算法的执行效率能够达到 50% 以上.

关键词: 防碰撞算法; Manchester 编码; RFID; 物联网

中图分类号: TP31

文献标识码: A

文章编号: 1000-7180(2011)07-0146-03

Skip Binary Tree Search Algorithm and its Analysis of Anti-collision

HONG Hai-liang, JIN Jie, JIAO Ting-ting

(1 School of Electronic and Information Engineering, Tianjin University, Tianjin 300072, China;

2 Institute of Economic and Social Development, Nankai University, Tianjin 300071, China)

Abstract: Proposed jump binary tree search anti-collision algorithm based on the introduction of the anti-collision algorithm in RFID system. LIFO stack by setting the full use of the collision node information to enable them to quickly and effectively identify RFID tags. Through its analysis, the efficiency of the algorithm can reach 50%.

Key words: anti-collision algorithm; Manchester encoding; RFID; internet of things

1 引言

物联网^[1]是在计算机互联网的基础上, 利用 RFID(Radio Frequency Identification)、无线数据通信等技术, 构造一个覆盖世界万物的“Internet of Things”. RFID 系统一般由电子标签和阅读器组成. 阅读器负责发送广播并接收标签的标识信息; 标签收到广播命令后将自身标识信息发送给阅读器. 然而由于阅读器与所有标签共用一个无线信道, 当阅读器识别区域内存在两个或者两个以上的标签在同一时刻阅读器发送标识信息时, 将产生碰撞, 导致阅读器不能对一些标签进行识别处理, 解决此碰撞的方法为防碰撞算法.^[2-3]

现在防碰撞算法有随机问询的 ALOHA 算法、时隙 ALOHA 算法, 信息的最佳利用率分别为 18.4%、36.8%, 但随着标签数量的扩大, 性能将急剧

恶化^[4].

文中在二进制树形搜索基本算法的基础上提出了基于跳跃式二进制树形搜索防碰撞算法, 该算法在执行的过程中将跳过空闲节点, 将能有效、快速地识别出区域内的所有标签.

2 算法约定

(1) 跳跃式二进制树形反碰撞算法的关键是确定数据发生碰撞的具体位置. 为了解决这一问题, 标签数据采用曼彻斯特编码^[5]调制. 该编码是用电平的变化(上升沿/下降沿)来表示数值位. 这里下降沿编码为逻辑“1”, 上升沿编码为逻辑“0”, 若无状态跳变, 视为非法数据, 作为错误被识别. 当两个或多个标签同时返回的数位有不同之值时, 则上升沿和下降沿互相抵消, 以至无状态跳变, 阅读器可知该位出现碰撞, 产生了错误, 应进一步搜索. 假设有 2 个 RFID 标签, 其 EPC 代码为 8 位, 利用 Manchester

编码能按位识别出碰撞位的示意图如图1所示。

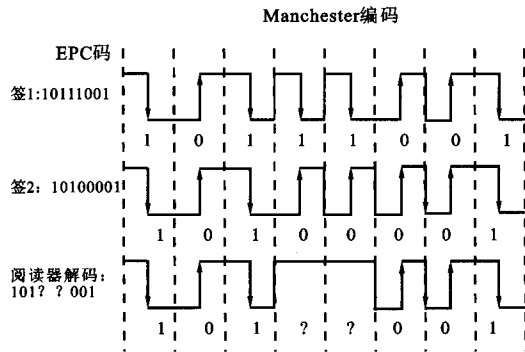


图1 用曼彻斯特编码的标签返回数据能够被识别出碰撞位

(2)为了便于算法执行的描述,定义两个命令 $\text{call}(\text{epc}, m)$ 及 $\text{sleep}(\text{epc}, m)$. 其中 $\text{call}(\text{epc}, m)$ 命令的含义为:读写器向覆盖范围内的标签发送召唤指令,如果标签 EPC 码与 call 命令中 epc 参数的前 m 位相等,那么满足这个条件的 EPC 标签作出应答,向读写器发送 EPC 码,不满足条件的 EPC 标签则处于空闲状态不作应答.特别地,在第一次 call 命令中 $\text{epc}=00000000, m=0$,作用范围内的所有标签都做出应答. $\text{sleep}(\text{epc}, m)$ 的含义与 call 命令类似,如果标签 EPC 码与 sleep 命令中 epc 参数的前 m 位相等,那么满足这个条件的 EPC 标签进入睡眠状态,再次收到 call 指令时不作应答.欲重新激活睡眠标签必须使得标签脱离读写器的作用范围才可奏效。

(3)定义一个 LIFO^[6] (Last In First Out) 栈用来存储算法执行过程中所经过的节点的信息。

3 算法原理

下面以一个实例来直观的描述该算法的原理.设在某时刻有四个标签同时进入读写器的作用范围内,它们的 EPC 码分别为 $\text{EPC1}=10100011, \text{EPC2}=10011011, \text{EPC3}=01010101, \text{EPC4}=11101100$. 如图2所示,其 call 命令的执行过程如下:

(1)发送 $\text{call}(00000000, 0)$ 命令, $\text{EPC1}, \text{EPC2}, \text{EPC3}, \text{EPC4}$ 均做出应答. 混叠的数据返回给读写器,由于此时各个位均有冲突,所以读写器识别为?????????. 接下来算法将 call 命令中的 epc 的最高冲突位置为 0, $\text{epc}=0???????,$ 然后根据 epc 参数求出 m , 此时 $m=1$. 同时算法将接收到的 epc 最高冲突位置为 1, 并保存在用于存储树形节点信息的 LIFO(last in first out)栈中。

(2)发送 $\text{call}(0???????, 1)$ 命令, 此时只有

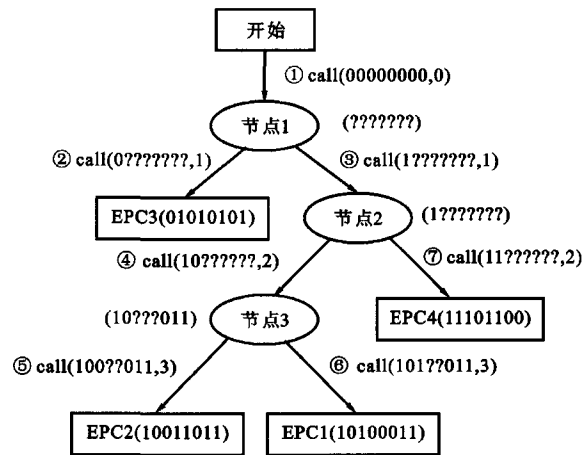


图2 跳跃式二进制树形防碰撞算法示意图

EPC3 满足 call 命令的应答条件,作出应答. 读写器收到的信息仅由 EPC3 发出,不存在碰撞问题. 读写器将 EPC3 的 EPC 码 01010101 存入 RAM 中等待处理. 然后发送 $\text{sleep}(01010101, 8)$ 命令,强制 EPC3 进入睡眠状态. 最后算法读出 LIFO 栈中的节点信息, $\text{epc}=1???????,$ 求出 $m=1$. 为下一次 call 命令做准备。

(3)发送 $\text{call}(1???????, 1)$ 命令, 此时 $\text{EPC1}, \text{EPC2}, \text{EPC4}$ 满足条件做出应答. 读写器识别为 1???????, 读写器接下来重复类似(1)中的操作,形成下一次 call 命令的参数 $\text{epc}=10???????, m=2$. 并将节点信息 11?????? 存入 LIFO 栈中。

(4)发送 $\text{call}(10???????, 2)$ 命令, 此时 $\text{EPC1}, \text{EPC2}$ 满足条件做出应答. 读写器识别为 10??? 011, 读写器将最高冲突位置 0, 形成下一次 call 命令的参数 $\text{epc}=100?? 011, m=3$. 并将最高冲突位置 1, 作为节点信息保存在 LIFO 栈中。

(5)发送 $\text{call}(100?? 011, 3)$ 命令, 此时只有 EPC2 满足 call 命令的条件做出应答, 读写器识别位 10011011, 不存在碰撞问题. 读写器将 EPC2 的 EPC 码 10011011 存入 RAM 中等待处理. 然后发送 $\text{sleep}(10011011, 8)$ 命令,强制 EPC2 进入睡眠状态. 最后算法读出 LIFO 栈中的节点信息, $\text{epc}=101?? 011,$ 求出 $m=3$. 为下一次 call 命令做准备。

(6)发送 $\text{call}(101?? 011, 3)$ 命令, 此时只有 EPC1 满足 call 命令的条件做出应答, 读写器识别位 10100011, 不存在碰撞问题. 读写器将 EPC1 的 EPC 码 10100011 存入 RAM 中等待处理. 然后发送 $\text{sleep}(10100011, 8)$ 命令,强制 EPC1 进入睡眠状态. 最后算法读出 LIFO 栈中的节点信息, $\text{epc}=11???????,$ 求出 $m=2$. 为下一次 call 命令做准备。

(7)发送 call(11?????,2)命令,此时只有 EPC4 满足 call 命令的条件做出应答,读写器识别位 11101100,不存在碰撞问题.读写器将 EPC4 的 EPC 码 11101100 存入 RAM 中等候处理.然后发送 sleep (11101100,8)命令,强制 EPC4 进入睡眠状态.此时 LIFO 栈为空,说明算法已经遍历了二进制树的每一个节点.至此整个反碰撞算法执行完毕,EPC1、EPC2、EPC3、EPC4 的 EPC 码已经被识别出来.

从以上的过程中可以看出,call 命令的 epc 参数由碰撞位的判断得出,而 call 命令中的 m 参数又由相应的 epc 参数求得的,这样就使得算法在执行过程中跳过了空闲的节点;又由于算法在执行过程中使用了堆栈,从而进一步提高了算法的执行效率.因为算法可以形象地用二进制树来描述,故称为“跳跃式二进制树形防碰撞算法”.

4 算法流程

算法流程如图 3 所示.

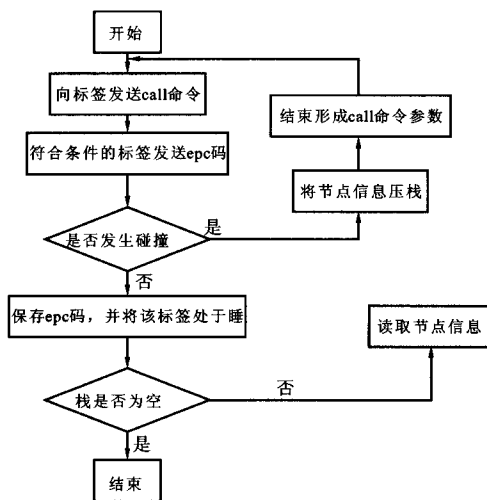


图 3 跳跃式二进制树形反碰撞算法的流程图

5 算法分析

对于形如图 2 所示的一个具有 N 个叶的二进制树,它的分支节点数目则为 $(N-1)$,每个分支节点具有两个分支,总的分支数目即为 $(2N-2)$,再加上主干后数目为 $(2N-1)$.对于本算法的二进制树主干与分支的数目总合即为读写器发送 call 命令的次数,二进制树叶的数目即为标签的数目.从而可以得出这样的结论:

分辨 N 个标签共需要 $(2N-1)$ 个 call 命令时

隙.算法的效率公式为

$$\eta = N / (2N - 1)$$

从图 4 可以看出,当 N 逐渐增大时算法的效率趋近于 50%.

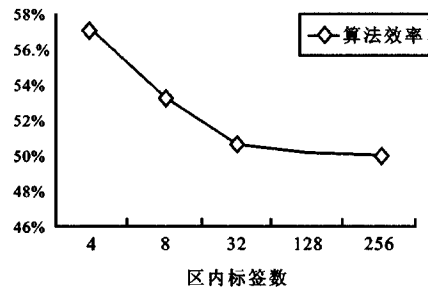


图 4 算法执行效率曲线图

6 结束语

文中提出了跳跃式二进制树形防碰撞算法,该算法在二进制搜索基本算法的基础上通过与堆栈的结合使其能快速、有效地识别出多个标签,这对于物联网中大批量物品的识别、追踪、管理具有极其重要的现实意义.

参考文献:

- [1] 宁焕生,王炳辉. RFID 重大工程与国家物联网[M]. 北京:机械工业出版社,2009.
- [2] Finkenzeller K. 射频识别(RFID)技术[M]. 陈大才,译. 北京:电子工业出版社,2006.
- [3] 吴博,周铜,王栋. RFID 防碰撞算法分析与研究[J]. 微电子学与计算机,2009,26(8):237-242.
- [4] Richard Kalinowski, Michel Latteux, David Simpwt. An adaptive Anti-collision Protocol for Smart Labels. [EBIOL]. [2010-03-13]http://www.jifl.fr/~simplot/recherch/articles,2001.
- [5] 余松森,詹宜巨,彭卫东,等. 基于后退式索引的二进制树形搜索反碰撞算法及其实现[J]. 计算机工程与应用,2004,40(16):26-28.
- [6] 戚桂杰. 数据结构[M]. 山东:山东人民出版社,2007.

作者简介:

洪海亮 男,(1986-),硕士生,研究方向为网络技术、智能仪器及控制.

金杰 女,(1962-),教授、博士生导师,研究方向为光通信器件、网络技术、嵌入式智能仪器及控制.

矫亭亭 女,(1987-),硕士生,研究方向物流与供应链管理、外包绩效.