基于分组机制的跳跃式动态二进制防碰撞算法

王亚奇 [1] 蒋国平 [1, 2]

摘 要 在射频识别技术 (Radio frequency identification, RFID) 系统中, 标签碰撞的解决对于标签的快速识别极为重要. 本文提出一种基于分组机制的跳跃式动态二进制防碰撞算法 (Anti-collision algorithm based on grouping mechanism and jumping dynamic binary, GJDB), 该算法通过在确定性算法中引入随机分组机制, 很好地解决了基于二进制搜索算法中, 由于标签数目不断增大所导致的识别效率降低的问题. 理论分析和仿真结果表明, GJDB 算法的性能优于其他常用的标签防碰撞算法, 并且该算法对标签随机分组数目的选取具有较强的鲁棒性.

关键词 射频识别, 标签碰撞, 二进制搜索, 鲁棒性

DOI 10.3724/SP.J.1004.2010.01390

Anti-collision Algorithm Based on Grouping Mechanism and Jumping Dynamic Binary

WANG Ya-Qi[1] JIANG Guo-Ping[1, 2]

Abstract In radio frequency identidication (RFID) systems, tag collision resolution is very important for fast tag identification. This paper presents an anti-collision algorithm based on grouping mechanism and jumping dynamic binary (GJDB). By introducing the randomized grouping mechanism into deterministic algorithm, the GJDB algorithm can solve the identification efficiency decrease of binary-based search algorithms caused by the continuously increasing number of tags. Theoretical analysis and simulation results show that the GJDB algorithm outperforms other tag anti-collision algorithms and has strong robustness against the randomized grouping number of tags.

Key words Radio frequency identification (RFID), tag collision, binary-based search, robustness

射频识别 (Radio frequency identification, RFID) 是一种利用无线射频方式在阅读器和标签之间进行非接触双向数据传输, 进而获取被标识物体信息的识别技术[1]. 该技术具有识别距离远、穿透能力强、多目标识别和抗污染等优点, 现已广泛应用于诸多领域. 借助于 RFID 技术, 可以构建全新的"物联网络". 在"物联网络" 中, 任何物体都可以利用 RFID 标签进行唯一标识, 成为网络中的一个节点, 从而能够自动与网络进行连接、交互, 并被追踪、定位等. 与传统网络相似, RFID 阅读器与标签之间的数据经过空间信道传输, 同样面临信道共享和访问冲突问题, 由于多个标签共享 Tag-to-reader 的上行信道, 当多个标签同时回应阅读器查询时, 如果没有相应的防碰撞机制, 必然会产生信道争用, 导致信号之间互相干扰, 即发生了标签碰撞[2]. 在 RFID 系统中, 防碰撞技术是信号识别和处理的关键技术之一.

现在被业界所推崇的防碰撞算法主要分为两类: 一类是基于 Aloha 协议的随机算法, 另一类是基于二进制树的确定算法. 基于 Aloha 协议的防碰撞算法一般包括时隙 Aloha 算法 (Slotted Aloha, SA)[3]、帧时隙 Aloha 算法 (Frame-slotted Aloha, FSA)[4]、动态帧时隙 Aloha 算法 (Dynamic frame-slotted Aloha, DFSA)[5−6] 和改进的 EPC Gen2 动态帧时隙 Aloha 算法 (Active slotted random, ASR)[7] 等; 基于二进制树的主要代表算法有动态二进制搜索 (Dynamic binary search, DBS) 算法[8]、跳跃式动态树 (Jumping and dynamic searching, JDS) 算法[9−11]、查询树改进 (Query tree improvement, QTI) 算法[12] 和 (Stack-based ID-binary tree, SIBT) 算法[13] 等. 这两类算法存在各自的优缺点: Aloha 算法的复杂度及对标签的要求较低, 但存在不稳定的工作区间[14], 并且可能导致 "标签饥饿问题 (Tag starvation problem)"[15], 即一些特定的标签可能在很长时间内都无法被识别; 二进制树算法的识别率可达 100 %, 即不存在 "标签

收稿日期 2009-05-25 录用日期 2010-06-22
Manuscript received May 25, 2009; accepted June 22, 2010
国家自然科学基金 (60874091), 国家教育部新世纪优秀人才支持计划 (NECT-06-0510), 江苏省高校自然基金基础研究计划 (08KJD510022) 资助
Supported by National Natural Science Foundation of China (60874091), the Program for New Century Excellent Talents in University (NCET-06-0510), and the Natural Science Basic Research Project for Universities of Jiangsu Province (08KJD510022)

1. 南京邮电大学控制与智能技术研究中心 南京 210003 2. 南京邮电大学自动化学院 南京 210003
1. Center for Control and Intelligence Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003 2. College of Automation, Nanjing University of Posts and Telecommunications, Nanjing 210003

饥饿问题", 但算法比较复杂, 识别时间较长.

结合 Aloha 算法和树形算法的优点, 文献 [16] 提出了 TSA (Tree slotted Aloha) 算法, 该算法通过及时处理碰撞标签提高了识别效率, 然而处理方法仍然采用的是动态帧时隙 Aloha 算法; EB-FSA (FSA with robust estimation and binary selection) 算法则利用二进制选择策略识别碰撞时隙中的标签, 从而获得了较高的识别效率[17], 但该算法在标签估算阶段存在浪费时隙的情况; FSAPB (FSA with small pilot frame and binary selection) 算法采用分组机制减少了 EB-FSA 算法在标签估算阶段浪费的时隙[18], 但在识别过程中确定附加时隙数的大小则比较困难, 这直接影响了 FSAPB 算法的识别效果; 最近, 文献 [19] 提出了 GDF-BTA (Grouped dynamic frame and binary tree recursive identification) 算法来进一步提高系统的识别效率, 然而该算法需要准确估算标签数目, 并且由于在冲突解决期 (Conflict resolution interval, CRI) 内标签以随机地址法进行分裂, 因此与传统网络中使用的确定地址分裂算法相比, 该算法的吞吐率依然偏低[20]; 此外, 阅读器的数据传输量也比较大.

针对上述研究所存在的问题与不足, 本文基于分组机制 (Grouping mechanism) 和跳跃动态二进制识别 (Jumping dynamic binary identification), 提出一种新的二进制树标签防碰撞算法 (Anti-collision algorithm based on grouping mechanism and jumping dynamic binary, GJDB), 该算法分为标签估算和标签识别两个阶段. 在标签估算阶段, 对标签随机分组后只执行一次估算操作, 完成对未识别标签数的估算; 在标签识别阶段, GJDB 算法根据估算所得的标签数, 确定对剩余各组标签进行第二次分组即最优分组的分组数, 同时对碰撞标签采用 JDS 算法加以识别. GJDB 算法的特点在于对标签随机分组数的选取具有较强的鲁棒性, 并且在识别每组标签的过程中, 阅读器根据检测到的数据碰撞比特位信息把标签分配到二叉树的左右子树, 能够获得较高的识别效率. 仿真结果显示, 与其他常用的标签防碰撞算法相比, GJDB 算法能够显著提高系统吞吐率, 降低阅读器和标签数据传输量; 此外, 该算法对标签数目的估算精确度要求不高, 并且其识别效率不受标签 ID 长度的影响.

本文其余部分安排如下: 第 1 节提出系统模型; 第 2 节给出 GJDB 算法的性能分析; 第 3 节给出数值仿真结果; 第 4 节总结全文.

1 系统模型

在识别标签的过程中, 由于 GJDB 算法采用 JDS 算法处理标签的碰撞, 本节首先简要概述 JDS 算法, 随后详述 GJDB 算法.

1.1 JDS 算法

JDS 算法把整个标签的识别过程分为前进搜索 (Forward search) 和后退搜索 (Backward search)[11]. 前进搜索过程和 DBS 算法相同, 后退搜索则是当阅读器成功识别标签后, 就回跳到该节点的父节点继续识别其余标签, 详细识别过程如表 1 所示 (其中 × 表示数据发生碰撞位). 识别 n 个标签, JDS 算法所需的搜索次数仅为 $2n-1$.

结合表 1 归纳 JDS 算法要点如下: 1) 阅读器发送 Request (NULL, N) 命令 (N 为标签 ID 长度), 要求查询区域内所有标签应答; 2) 检测有无碰撞发生, 若有把最高碰撞位置 0, 高于该位的数值不变可得 ID_{N-1-x} 的值 (x 为 ID 中最高碰撞位的下标), 即为防碰撞命令参数 UID[8], 由此求出下一次查询命令所需的两个参数; 3) 若无碰撞则识别单个标签. 处理完后回跳到父节点, 得到下一次查询命令所需的两个参数; 4) 重复进行请求与检测过程, 直到执行 Request (Null, N) 命令无碰撞发生时结束.

表 1 JDS 算法的搜索过程

Table 1 The searching process of JDS algorithm

Forward search	1st search	2nd search	Backward search	3rd search	Forward search	4th search	5th search	Backward search	6th search	Backward search	7th search
Request	(Null, 8)	(10, 6)		(Null, 8)		(1110, 4)	(11100, 3)		(1110, 4)		(Null, 8)
Received code	1×1××010	Select Tag 1		111××010		×010	Select Tag 3		Select Tag 4		Select Tag 2
Tag 1	10110010	110010									
Tag 2	11110010			11110010							11110010
Tag 3	11100010			11100010		0010	010				
Tag 4	11101010			11101010		1010			1010		

1.2 GJDB 算法

为了便于描述 GJDB 算法, 引入标签分组命令, 该命令包括随机分组命令 Request(M, a) 和最优分组命令 Request(p, b) 两种形式, 其中, M 和 p 分别为随机分组和最优分组的分组数目参数, 并且 $a \in [1, M]$, $b \in [1, p]$.

1) Request(M, a) 命令实现的功能是阅读器要求查询区域内所有标签产生一个 $[1, M]$ 之间的随机数 (约定满足均匀分布, 下同), 并且选择随机数 a 的标签回传其 ID, 而选择其他随机数的标签被暂时 "屏蔽", 直到阅读器发送下一个 Request(M, a) 或 Request(M, 0) 命令时被 "唤醒", 这里的 Request(M, 0) 命令约定为阅读器要求查询区域内所有剩余标签回传其 ID.

2) Request(p, b) 命令的功能为阅读器要求发生碰撞的标签重新产生一个 [1, p] 之间的随机数, 选择随机数 b 的标签回传其 ID, 并暂时 "屏蔽" 产生其他随机数的标签, 直到阅读器发送下一个 Request(p, b) 或 Request(p, 0) 命令时被 "唤醒", 这里约定命令 Request(p, 0) 为阅读器要求某一组内所有剩余标签回传其 ID.

在 GJDB 算法中采用 Manchester 编码, 以便能准确地检测出阅读器中数据发生碰撞比特位的位置[8], 并约定阅读器作用范围内的标签能在同一时刻开始传送数据.

1.2.1 标签估算阶段

阅读器首先发送一个标签随机分组命令 Request(M, 1), 所有处于阅读器查询范围内的标签都产生一个 $[1, M]$ 之间的随机数, 这里把产生相同随机数的标签分为一组, 则所有标签就被随机分成了 M 组, 随后产生随机数 1 的标签向阅读器传送其 ID. 如果没有检测到碰撞发生, 阅读器成功识别该标签; 若发生了碰撞, 阅读器则发送 Request(p, 1) 和 Request(p, 0) 命令对碰撞标签进行识别, 并在识别过程中记录下已经识别的标签数目, 具体识别过程见第 1.2.2 节, 这里假设 p 的取值为 $p = M$. 若识别第 "1" 组标签后阅读器记录的标签数目为 n, 并假设每组中的标签数目都服从均匀随机分布, 则总的标签数目约为 Mn . 当 M 太大时, 每组中的标签数目可能不再服从均匀随机分布, 因此适当的 M 可以根据实际应用环境中标签规模的大小由人工调整设定. 即使 M 选取的过大以至于每组中的标签数目不再满足均匀随机分布, GJDB 算法仍然能够获得较高的识别效率, 因为该算法对标签数的估算精度要求不高, 并且对 M 的选取具有较强的鲁棒性 (见第 3 节).

1.2.2 标签识别阶段

在标签估算阶段, 阅读器已经识别出第 "1" 组的 n 个标签, 在标签识别阶段, 阅读器将分别识别剩余的 $M-1$ 组共 $(M-1)n$ 个标签. 由于随机数产生过程相互独立, 因此不同时刻标签产生 [1, p] 之间任意一个随机数的概率相同. 在识别剩余 $(M-1)n$ 个标签的过程中, 阅读器只需依次发送 $M-2$ 次 Request(M, 1) 命令和一次 Request(M, 0) 命令, 就能确保剩余标签重新被划分为 $M-1$ 组并依次向阅读器传送其 ID, 而每组标签数目依然满足均匀分布, 即每组标签数目都约为 n. 最后发送一次 Request(M, 0) 命令要求所有剩余标签而不仅是产生随机数 1 的标签响应, 目的就是确保识别率为 100 %. 发送 Request(M, 1) 命令或 Request(M, 0) 命令后, 若有标签响应并检测到碰撞发生, 阅读器随即发送最优分组命令 Request(p, 1) 对该组中的标签进行第二次分组, 即最优分组. 同理, 阅读器依次发送 $p-1$ 次 Request(p, 1) 命令和一次 Request(p, 0) 命令就可以把该组内的标签再次分为 p 组, 并依次向阅读器发送其 ID, 若检测到标签碰撞则利用 JDS 算法进行处理, 这里令 $p = \gamma n = [0.6n]$ ($[\cdot]$ 表示取整), 因为当 $p = \gamma n = [0.6n]$ 时 GJDB 算法的识别性能最优 (证明见第 2.1 节). GJDB 算法详细描述如下:

步骤 1. 阅读器发送标签随机分组命令 Request(M, 1), 要求查询区域内剩余的所有标签产生一个 [1, M] 之间的随机数, 产生随机数 1 的标签向阅读器传送其 ID, 并且暂时 "屏蔽" 选择其他随机数的标签, 直到阅读器发送下一个 Request(M, 1) 或 Request(M, 0) 命令时被 "唤醒".

步骤 2. 阅读器检测有无标签响应, 若无响应转到步骤 10; 若有响应, 检测有无标签碰撞发生.

步骤 3. 若无标签发生碰撞, 阅读器直接识别该标签, 随后转到步骤 10.

步骤 4. 若有碰撞发生, 阅读器发送最优分组命令 Request(p, 1), 发生碰撞的标签重新产生一个 [1, p] 之间的随机数, 选择随机数 1 的标签回传其 ID, 并暂时 "屏蔽" 产生其他随机数的标签, 直到阅读器发送下一个 Request(p, 1) 或 Request(p, 0) 命令时被 "唤醒".

步骤 5. 阅读器检测有无标签响应, 若无响应转到步骤 8; 若有响应, 检测有无标签碰撞发生.

步骤 6. 若无标签发生碰撞, 阅读器成功识别该标签, 随后转到步骤 8.

步骤 7. 若发生了标签碰撞, 阅读器则利用 JDS 算法来识别发生碰撞的标签.

步骤 8. 转到步骤 4 后顺序执行, 并循环 $p-1$ 次.

步骤 9. 在第 p 次时, 阅读器发送请求命令 Request(p, 0), 要求该组中所有剩余标签响应, 随后转到步骤 5, 并顺序执行到步骤 7 后转到步骤 10.

步骤 10. 转到步骤 1 后顺序执行, 并循环 $M-2$ 次.

步骤 11. 在第 $M-1$ 次时, 阅读器发送请求命令 Request(M, 0), 要求整个识别范围内所有剩余标签响应, 随后转到步骤 2, 并顺序执行到步骤 9 后可识别所有标签, 则整个识别过程结束.

上述识别过程也可用图 1 的流程框图来描述. 从上述识别过程可知, 标签识别阶段的核心是对剩余 $M-1$ 组中的每组标签再次执行分组操作, 即阅读器发送 $p-1$ 次 Request(p, 1) 命令和一次 Request(p, 0) 命令, 把每组中的 n 个标签再次随机分为 p 组来分别识别, 并借助于 JDS 算法对碰撞标签进行处理, 这与标签估算阶段识别第 "1" 组标签的过程相同, 不同的是在标签识别阶段, 对剩余的每组标签进行了最优分组, 即 p 取得最优值 $p=[0.6n]$, 而不是 M.

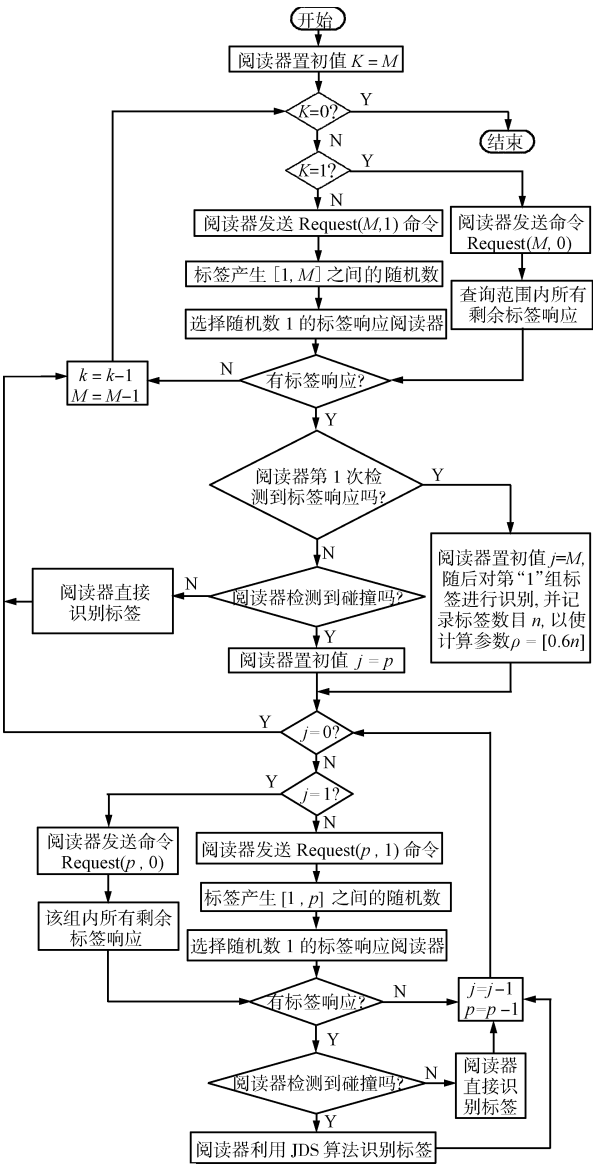

图 1　GJDB 算法的流程框图

Fig. 1　The flow chart of GJDB

2　GJDB 算法性能分析

本节主要讨论两个重要的性能指标, 一个是阅读器发送的搜索次数, 它决定了 CRI 的识别延迟; 另一个是通信复杂度, 一般用阅读器发送的总数据量 B_R 以及单个标签传送的平均数据量 B_T 来衡量, 它同时影响识别延迟以及阅读器和标签的功耗. 讨论过程中约定阅读器查询范围内的标签数目为 Mn, 标签 ID 的长度为 N. 为了便于分析问题而又不失一般性, 我们把这 Mn 个标签均匀分成 M 组.

2.1　搜索次数

1) 在标签估算阶段, GJDB 算法首先识别出 M 组中第 "1" 组的 n 个标签; 在标签识别阶段, GJDB 算法分别识别剩余的 $M-1$ 组标签. 当识别 M 组中任意一组标签时, 因为不同时刻每个标签选择 $[1, p]$ 之间任何一个随机数的概率相同, 因此识别过程等效于在同一时刻, 该组中每个标签都随机产生一个 $[1, p]$ 之间的随机数, 然后阅读器再依次发送参数 $1, 2, \cdots, p-1, p$, 标签用其产生的随机数与接收到的参数进行比较, 相符则回传其 ID. 令 $S_{\mathrm{JDS}}(n)$ 为利用 JDS 算法识别 n 个标签时阅读器的搜索次数, 由于选择 $[1, p]$ 之间任何一个随机数的标签数目服从二项分布, 则识别剩余任意一组中的 n 个标签所需阅读器的平均搜索次数 $S_{\mathrm{GJDB}}(n)$ 为

$$S_{\mathrm{GJDB}}(n)=1+\sum_{i=1}^{n}\left(\binom{n}{i}\left(\frac{1}{p}\right)^{i}\left(1-\frac{1}{p}\right)^{n-i}pS_{\mathrm{JDS}}(i)\right)+p\left(1-\frac{1}{p}\right)^{n} \tag{1}$$

式 (1) 中等号右边的第 1 项表示, 识别某一组标签时阅读器首先发送的一个随机分组命令 Request(M, 1) 或 Request(M, 0); 第 2 项表示, 阅读器用 JDS 算法对产生 $[1, p]$ 中任意一个随机数的碰撞标签分别进行识别所需的搜索次数; 最后一项是指阅读器发送的没有标签响应的 Request(p, 1) 或 Request(p,

0) 命令次数. 式 (1) 可进一步化简为

$$
\begin{aligned}
&S_{\mathrm{GJDB}}(n)=\\
&\sum_{i=1}^{n}\left(\binom{n}{i}\left(\frac{1}{p}\right)^{i}\left(1-\frac{1}{p}\right)^{n-i}p(2i-1)\right)+\\
&p\left(1-\frac{1}{p}\right)^{n}+1=\\
&\sum_{i=0}^{n}\left(\binom{n}{i}\left(\frac{1}{p}\right)^{i}\left(1-\frac{1}{p}\right)^{n-i}p(2i-1)\right)+\\
&2p\left(1-\frac{1}{p}\right)^{n}+1=\\
&2\sum_{i=0}^{n}\left(\binom{n}{i}\left(\frac{1}{p}\right)^{i-1}\left(1-\frac{1}{p}\right)^{n-i}i\right)-\\
&p\sum_{i=0}^{n}\left(\binom{n}{i}\left(\frac{1}{p}\right)^{i}\left(1-\frac{1}{p}\right)^{n-i}\right)+\\
&2p\left(1-\frac{1}{p}\right)^{n}+1=\\
&2\sum_{i=1}^{n}\left(\binom{n-1}{i-1}\left(\frac{1}{p}\right)^{i-1}\left(1-\frac{1}{p}\right)^{n-i}n\right)-\\
&p\sum_{i=0}^{n}\left(\binom{n}{i}\left(\frac{1}{p}\right)^{i}\left(1-\frac{1}{p}\right)^{n-i}\right)+\\
&2p\left(1-\frac{1}{p}\right)^{n}+1=\\
&2n\left(\frac{1}{p}+1-\frac{1}{p}\right)^{n-1}-p\left(\frac{1}{p}+1-\frac{1}{p}\right)^{n}+\\
&2p\left(1-\frac{1}{p}\right)^{n}+1=2n-p+2p\left(1-\frac{1}{p}\right)^{n}+1
\end{aligned}
\tag{2}
$$

为了求出 p 的最佳值 p_{opt}, 令 $p=\gamma n$ $(\gamma>0)$, 并把 $p=\gamma n$ 代入式 (2) 可得

$$
S_{\mathrm{GJDB}}(n)=2n-\gamma n+2\gamma n\left(1-\frac{1}{\gamma n}\right)^{n}+1 \tag{3}
$$

把式 (3) 两边同除以 n, 考虑 $n\to\infty$ 时的极限可得

$$
\begin{aligned}
&\lim_{n\to\infty}\frac{S_{\mathrm{GJDB}}(n)}{n}=\\
&\lim_{n\to\infty}\left(2-\gamma+2\gamma\left(1-\frac{1}{\gamma n}\right)^{n}+\frac{1}{n}\right)=\\
&2-\gamma+2\gamma\mathrm{e}^{-\frac{1}{\gamma}}
\end{aligned}
\tag{4}
$$

当 $\gamma\approx 0.6$ 时, 式 (4) 的右边取得最小值. 因此, 当 $p_{\mathrm{opt}}=\gamma n=0.6n$ 时, 本文提出的 GJDB 算法在识别 n 个标签时所用搜索次数最少, 算法识别性能达到最优. 由此可知, 利用 GJDB 算法识别 Mn 个标签所需阅读器的平均搜索次数 $S_{\mathrm{GJDB}}\left(Mn\right)$ 为

$$
\begin{aligned}
&S_{\mathrm{GJDB}}(Mn)=\\
&\sum_{i=1}^{n}\left(\binom{n}{i}\left(\frac{1}{M}\right)^{i}\left(1-\frac{1}{M}\right)^{n-i}MS_{\mathrm{JDS}}(i)\right)+\\
&M\left(1-\frac{1}{M}\right)^{n}+1+(M-1)\left(\sum_{i=1}^{n}\left(\binom{n}{i}\times\right.\right.\\
&\left.\left.\left(\frac{1}{p_{\mathrm{opt}}}\right)^{i}\left(1-\frac{1}{p_{\mathrm{opt}}}\right)^{n-i}p_{\mathrm{opt}}S_{\mathrm{JDS}}(i)\right)+\right.\\
&\left.p_{\mathrm{opt}}\left(1-\frac{1}{p_{\mathrm{opt}}}\right)^{n}+1\right)
\end{aligned}
\tag{5}
$$

2) GJDB 算法的优化. 识别 M 组标签的过程中, 若产生 [1, p] 之间同一个随机数的两个标签的 ID 中只有一位不同, 利用二进制位上取值非 0 即 1 的特性, 阅读器只需发送一次 Request 命令就可以同时识别这两个标签. 例如: 阅读器发送一次 Request 命令后, 若检测到的代码是 1110×010, 则可以直接判断出发生碰撞的两个标签的 ID 分别为 11100010 和 11101010, 因此阅读器无需再发送 Request 命令就可以直接识别这两个标签. 这样阅读器只需发送一次查询命令就可以识别两个标签, 而通常情况下使用 JDS 算法阅读器需要发送 3 次查询命令. 因此在 GJDB 算法中, 利用标签 ID 中二进制位上取值非 0 即 1 的特性可以减少阅读器的搜索次数, 即阅读器每识别两个具有上述特性的标签就可以减少两次查询.

为了方便分析, 我们假设标签的 N 位 ID 中若有 m 位发生碰撞, 标签的数目则为 2^m. 对于这 2^m 个标签中的任意一个标签, 有且仅有 m 个标签的 ID 与之发生碰撞的位数为 1. 因为产生 [1, p] 之间任意一个随机数的标签数目服从二项分布, 当产生同一随机数的两个标签的 ID 中仅有最低碰撞位不同的

概率 P_0 为

$$P_0 = \frac{1}{2}\sum_{i=0}^{2^m-2^0-1}\left(\begin{pmatrix}2^m\\1\end{pmatrix}\begin{pmatrix}2^m-2^0-1\\i\end{pmatrix}\times\left(\frac{1}{p}\right)^{2+i}\left(1-\frac{1}{p}\right)^{2^m-2-i}\right) \tag{6}$$

当产生同一随机数的两个标签的 ID 中仅有次低碰撞位不同的概率 P_1 为

$$P_1 = \frac{1}{2}\sum_{i=0}^{2^m-2^1-1}\left(\begin{pmatrix}2^m\\1\end{pmatrix}\begin{pmatrix}2^m-2^1-1\\i\end{pmatrix}\times\left(\frac{1}{p}\right)^{2+i}\left(1-\frac{1}{p}\right)^{2^m-2-i}\right) \tag{7}$$

同理可得, 当产生同一随机数的两个标签的 ID 中仅有最高碰撞位不同的概率 P_{m-1} 为

$$P_{m-1} = \frac{1}{2}\sum_{i=0}^{2^m-2^{m-1}-1}\left(\begin{pmatrix}2^m\\1\end{pmatrix}\begin{pmatrix}2^m-2^{m-1}-1\\i\end{pmatrix}\times\left(\frac{1}{p}\right)^{2+i}\left(1-\frac{1}{p}\right)^{2^m-2-i}\right) \tag{8}$$

在阅读器识别这 2^m 个标签的过程中, 具有上述特性的标签数目为

$$n_{\mathrm{GJDB}}(2^m) = 2\sum_{j=0}^{m-1}\left(\sum_{i=0}^{2^m-2^j-1}\left(\begin{pmatrix}2^{m-1}\\1\end{pmatrix}\begin{pmatrix}2^m-2^j-1\\i\end{pmatrix}\times\left(\frac{1}{p}\right)^{2+i}\left(1-\frac{1}{p}\right)^{2^m-2-i}\right)p\right) \tag{9}$$

若实际响应阅读器查询的标签数目不是 2^m 而是 Mn, 并假设每个标签的 ID 随机产生. 在阅读器识别这 Mn 个标签的过程中, 具有上述特性的平均标签数目是

$$n_{\mathrm{GJDB}}(Mn) = \frac{Mn}{2^{m-1}}\sum_{j=0}^{m-1}\left(\sum_{i=0}^{2^m-2^j-1}\left(\begin{pmatrix}2^{m-1}\\1\end{pmatrix}\begin{pmatrix}2^m-2^j-1\\i\end{pmatrix}\times\left(\frac{1}{p}\right)^{2+i}\left(1-\frac{1}{p}\right)^{2^m-2-i}\right)p\right) \tag{10}$$

因此, 利用标签的 ID 中二进制位上取值非 0 即 1 的特性, 当识别 Mn 个标签时, 可以把 GJDB 算法的搜索次数减少为

$$S'_{\mathrm{GJDB}}(Mn) = S_{\mathrm{GJDB}}(Mn) - n_{\mathrm{GJDB}}(Mn) \tag{11}$$

2.2 通信复杂度 B_R

GJDB 算法中阅读器发送的总数据量 B_R 包括 3 个部分: 1) 阅读器识别 Mn 个标签的过程中, 需要发送 $M-1$ 次 Request(M, 1) 命令和一次 Request(M, 0) 命令, 此时阅读器发送的数据量为 $M\left(\sum_{i=1}^{k}\left(i2^{i-1}+\left(M-2^k\right)(k+1)\right)+1\right)-1$, 其中, $k=\left\lfloor\log_2^{M-1}\right\rfloor$($\lfloor\cdot\rfloor$ 表示下取整); 2) 当阅读器识别剩余 $M-1$ 组标签中的任意一组标签时, 需要发送 $p-1$ 次 Request(p, 1) 命令和一次 Request(p, 0) 命令, 在这个过程中阅读器传输的数据量为 $p\left(\sum_{i=1}^{j}\left(i2^{i-1}+(p-2^j)(j+1)\right)+1\right)-1$, 其中 $j=\left\lfloor\log_2^{p-1}\right\rfloor$; 3) 阅读器用 JDS 算法识别发生碰撞标签时发送的数据量, 首先我们计算阅读器每次发送参数 UID 的平均长度如下.

在 JDS 算法中, 阅读器采用动态方式发送 UID 参数[9], 即阅读器每次发送的前缀码为标签 ID 中的一部分, 标签则发送其 ID 中除去前缀码后剩余的一串比特位. 在识别 Mn 个标签的过程中, 若标签的 N 位 ID 中有 m 位发生了碰撞, 并且假设碰撞发生的位置是随机的, 则标签的 N 位 ID 中某一位发生碰撞的概率为

$$P_{\mathrm{col}} = \frac{\begin{pmatrix}N-1\\m-1\end{pmatrix}}{m\begin{pmatrix}N\\m\end{pmatrix}} \tag{12}$$

因此在 JDS 算法中, 阅读器每次发送的 Request 命令所包含参数 UID 的平均长度 L_{UID} 为

$$L_{\mathrm{UID}} = \sum_{k=1}^{N}kP_{\mathrm{col}} = \sum_{k=1}^{N}k\frac{\begin{pmatrix}N-1\\m-1\end{pmatrix}}{m\begin{pmatrix}N\\m\end{pmatrix}} = \sum_{k=1}^{N}k\frac{1}{N} = \frac{1+N}{2} \tag{13}$$

综上所述, 可得 GJDB 算法的通信复杂度 B_R 的表达式为[21]

$$B_R = (M-1)\times \left(p\left(\sum_{i=1}^{j}\left(i2^{i-1}+\left(p-2^j\right)(j+1)\right)+1\right)-1\right)+ M\left(\sum_{i=1}^{k}\left(i2^{i-1}+\left(M-2^k\right)(k+1)\right)+1\right)- 1+\left(S'_{\mathrm{GJDB}}(Mn)-M-(M-1)\,p\right)\times L_{\mathrm{UID}} \tag{14}$$

2.3 通信复杂度 B_T

为求 GJDB 算法的通信复杂度 B_T, 首先计算当利用 JDS 算法识别 Mn 个标签时, 响应阅读器查询的总标签数目.

引理 1. JDS 算法识别标签的整个过程为一个二叉树结构, 所有响应阅读器查询的标签数目可表示为二叉树中各个节点值的总和.

证明. 识别过程中, 如果把标签碰撞视为分支节点, 标签正确识别为叶子结点, 又因为标签的 ID 用二进制表示, 则这个结构的分支不会大于 2, 故整个识别过程必定是二叉树结构. 在识别标签的过程中, 阅读器根据检测到的数据碰撞比特位信息把标签分配到二叉树的左右子树, 因此, 可以用节点值表示响应该查询命令的标签数目, 则二叉树中各个节点值的总和即为整个识别过程中所有响应阅读器查询的标签数目. □

举例对引理 1 进行说明, 表 1 中标签响应阅读器查询的过程可以表示为图 2 的形式. 如图 2 所示, 识别 4 个标签, 阅读器发送了 7 次查询命令, 分别对应于二叉树中的 7 个节点, 其中, 3 个分支节点表示发生了三次标签碰撞, 4 个叶子节点表示四次标签正确识别, 而节点值则是每次响应阅读器查询的标签数目, 所以阅读器识别这 4 个标签的过程中, 发送响应信号的标签数目为 13.

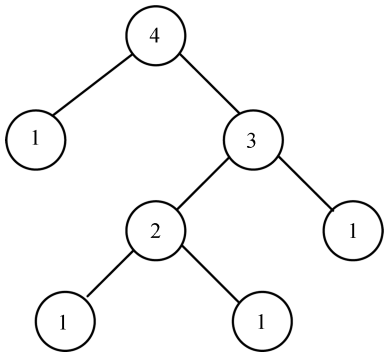

图 2 标签的响应过程

Fig. 2 The response process of tag

由引理 1 可知, 当用 JDS 算法识别 Mn 个标签时, 为了求出响应阅读器查询的标签数目, 可以把这 Mn 个标签按二叉树进行分解, 然后对各个节点值求和即可. 下面首先考虑两种特殊的情况 (分别对应于定理 1 和定理 2): 1) Mn 个标签的 ID 中 m 个碰撞位所表示的二进制数从 0 开始, 顺序递增直到 2^m 为止, 此时 $Mn = 2^m$; 2) 把 Mn 个标签的 ID 中 m 个碰撞位所表示的二进制数从小到大进行排列, 最小二进制数中的最高碰撞位即第 $m-1$ 位应当为 0, 第二个数仅有最高位为 1, 其余 $m-1$ 个碰撞位均为 0, 第三个数仅有最高位和次高位为 1, 其余 $m-2$ 个碰撞位均为 0, 以此类推最后一个数的 m 个碰撞位均为 1.

定理 1. 利用 JDS 算法识别 Mn 个标签时, 响应阅读器查询的最小标签数目为 $Mn\,(m+1)$.

证明. 为了求出响应阅读器查询的最小标签数目, 根据引理 1, 应当把 Mn 个标签按照总节点值之和最小的方式进行分解, 而各个节点值之和最小的分解方式为对等分解, 如图 3 所示. 在这种情况下构造的二叉树即为层数为 $\left[\log_2^{Mn}\right]+1$ 的满二叉树, 而每层内的标签数目均为 Mn, 由此求出阅读器识别 Mn 个标签时, 响应阅读器查询的最小标签数目为 $Mn\left(\left[\log_2^{Mn}\right]+1\right)$, 即为 $Mn\,(m+1)$. □

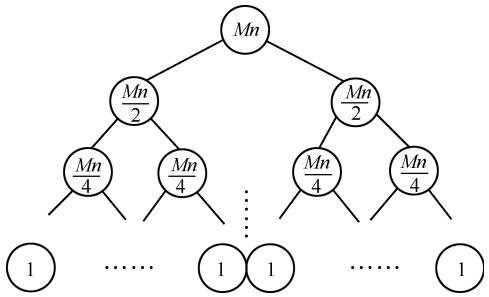

图 3 最优情况

Fig. 3 The best case

定理 2. 利用 JDS 算法识别 Mn 个标签时, 响应阅读器查询的最大标签数目为 $Mn+\frac{1}{2}Mn\,(Mn+1)-1$.

证明. 由引理 1 可知, 为了求出响应阅读器查询的最大标签数目, 应当把 Mn 个标签按照总的节点值之和最大的方式进行分解. 该分解方法为 (每层节点没有先后次序): Mn 分解为 1 和 $Mn-1$, $Mn-1$ 进一步分解为 1 和 $Mn-2$, 以此类推直到把 2 分解为 1 和 1 为止, 分解过程如图 4 所示. 这种分解方式构造的二叉树的层数为 Mn , 而每层的标签数从根节点到叶子节点依次为 $Mn, Mn, Mn-1, \cdots, 3, 2$, 若 Mn 个标签按照这种方式分解, 则可求出响应阅读器查询的最大标签数目为 $Mn+\frac{1}{2}Mn\,(Mn+1)-1$. □

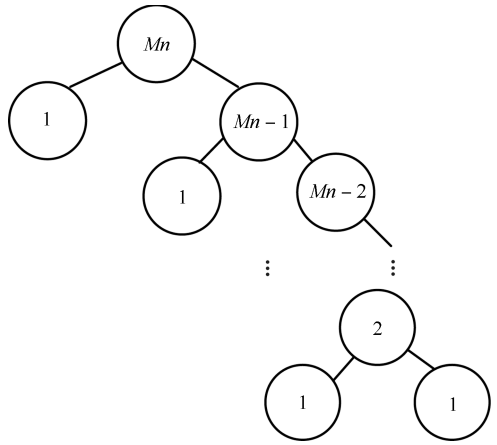

图 4　最坏情况

Fig. 4　The worst case

在利用 JDS 算法识别 Mn 个标签的过程中, 如果标签的 ID 是随机产生的, 则向阅读器发送响应信号的标签数目的取值范围为

$$Mn\,(m+1) < R_{\mathrm{JDS}}(Mn) < Mn + \frac{Mn(Mn+1)}{2} - 1 \tag{15}$$

对于 GJDB 算法, 在识别 Mn 个标签的过程中, 每个标签首先响应一次 Request(M, 1) 命令或 Request(M, 0) 命令, 随后依次响应 $p-1$ 次 Request(p, 1) 命令和一次 Request(p, 0) 命令, 并且阅读器利用 JDS 算法对发生碰撞的标签进行识别, 同时考虑 GJDB 算法的优化, 所以识别 Mn 个标签时, 响应阅读器查询的平均标签数目为

$$\begin{aligned} R_{\mathrm{GJDB}}(Mn) = & \sum_{i=1}^{n}\left(\binom{n}{i}\left(\frac{1}{M}\right)^{i}\left(1-\frac{1}{M}\right)^{n-i} MR_{\mathrm{JDS}}(i)\right) + \\ & 1 - R_{\mathrm{JDS}}(n_{\mathrm{GJDB}}(Mn)) + \\ & (M-1)\left(\sum_{i=1}^{n}\left(\binom{n}{i}\left(\frac{1}{p_{\mathrm{opt}}}\right)^{i} \times \left(1-\frac{1}{p_{\mathrm{opt}}}\right)^{n-i} p_{\mathrm{opt}} R_{\mathrm{JDS}}(i)\right) + 1\right) \end{aligned} \tag{16}$$

由第 2.2 节可知, 在 GJDB 算法中, 标签每次发送数据的平均长度为

$$\begin{aligned} L_{\mathrm{tag}} = \sum_{k=1}^{N} kP_{\mathrm{col}} = \sum_{k=1}^{N} k\frac{\binom{N-1}{m-1}}{m\binom{N}{m}} = \\ \sum_{k=1}^{N} k\frac{1}{N} = \frac{1+N}{2} \end{aligned} \tag{17}$$

由此可得 GJDB 算法的通信复杂度 B_T 的表达式为

$$B_T = \frac{R_{\mathrm{GJDB}}(Mn)L_{\mathrm{tag}}}{Mn} \tag{18}$$

3　仿真结果

本节选取 TSA、ASR 和 GDF-BTA 三种常用算法与 GJDB 算法进行比较. 考虑的性能指标包括识别延迟和通信复杂度, 其中识别延迟利用阅读器发送的搜索次数来衡量, 而通信复杂度依然用阅读器发送的总数据量 B_R 以及单个标签传送的平均数据量 B_T 来反映. 不同算法的读周期长度也各不相同, 为具有可比性, 本节用系统吞吐率来表示识别延迟, 这里的吞吐率定义为标签数目与识别其所需阅读器发送的搜索次数或总时隙数之比. 仿真环境如下: 仿真平台为 Matlab 7.6, 在阅读器查询范围内标签数目的取值范围为 100 ∼ 1 000, 标签的 ID 随机产生, 为考察其长度对 GJDB 算法的影响, N 依次取 48、96 和 128; 为说明 GJDB 算法对标签随机分组数的选取具有较强鲁棒性, M 的取值分别为 10、15、20、25 和 30, 并假设 20 为其初始值; GJDB 算法识别碰撞标签时的仿真条件同 JDS 算法[11], ASR 算法的仿真条件参考文献 [7], TSA 和 GDF-BTA 算法中的初始帧长设定为 128, 其他参数的选取可分别参考文献 [16] 和文献 [19]. 所有仿真结果均为 50 次独立运行后的平均值.

图 5 给出了在 $N=48$、$M=20$ 的情况下, GJDB 算法与 TSA、ASR 和 GDF-BTA 三种算法的吞吐率比较. 图 5 表明, GJDB 算法的吞吐率明显高于其他三种算法, 尤其当 $\gamma=0.6$ 时, 对于标签数目较多的应用环境, GJDB 算法的吞吐率能够达到 0.61, 此时的系统识别时延最小, 这是由于 GJDB 算法采取的最优分组策略大幅减少了每次响应阅读器查询的标签数目, 从而降低了标签发生碰撞的概率. 当对标签数目的估算存在偏差时 ($\gamma=0.3, 1.2$), 即 p 的取值并不是最优时, GJDB 算法的吞吐率最终可分别稳定在 0.57 和 0.54, 依然高于其他三种算法, 这表明 GJDB 算法对标签数目的估算精度要求不高. 由于采用了对标签动态分组后再利用二叉树

递归进行识别的策略, GDF-BTA 算法的吞吐率优于 TSA 和 ASR 算法, TSA 算法能够对碰撞标签及时进行处理, 从而获得了略好于 ASR 算法的识别性能.

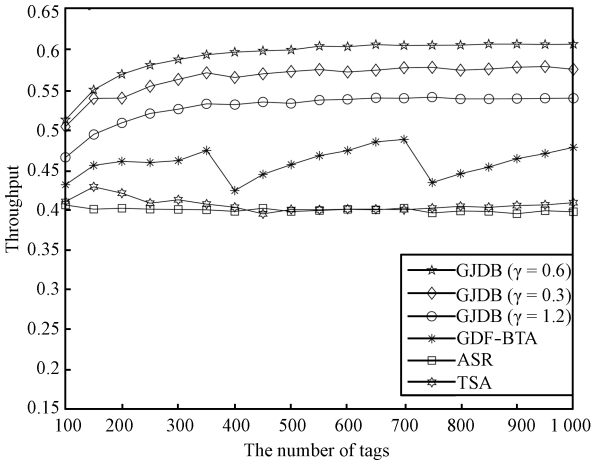

图 5 吞吐率的比较

Fig. 5 Comparison of throughputs

图 6 显示了当 $M = 20$、$\gamma = 0.6$ 时, 标签 ID 长度 N 对 GJDB 算法吞吐率的影响. 图 6 表明当标签 ID 长度发生变化时, GJDB 算法的吞吐率保持稳定, 这主要因为 GJDB 算法是根据 ID 中发生碰撞的二进制位信息来识别碰撞标签的, 因此, 在标签数量一定的情况下, GJDB 算法的吞吐率将会受到 ID 中发生碰撞二进制位位数的影响, 而与 ID 的长度并没有直接关系.

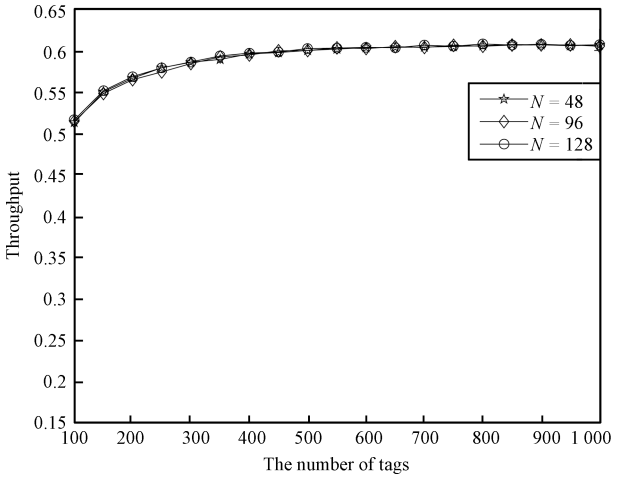

图 6 不同标签 ID 长度下吞吐率的变化

Fig. 6 Variation of throughputs under different lengths of tag′s ID

图 7 描述了在 $N = 48$、$\gamma = 0.6$ 的情况下, 标签随机分组数 M 取不同值时 GJDB 算法吞吐率的变化. 由图 7 可知, 当 M 的取值由初始值 $M = 20$ 分别变化为 $M = 10, 25, 30$ 时, 若标签数量为 100, GJDB 算法吞吐率的改变量分别为 11.7%、6.6%、6.8% 和 10.6%, 然而随着标签数量的增大, 吞吐率的改变量逐渐减小并趋于零, 这说明在 M 取值变化比较大 (10～30) 的情况下, 对于目前大规模标签的应用环境, GJDB 算法的吞吐率几乎不发生变化, 依然能够保持较好的识别性能. 由此可知, GJDB 算法对标签随机分组数 M 的选取具有较强的鲁棒性, 这是因为 GJDB 算法吞吐率的大小主要由标签最优分组数 p 来决定, 而受随机分组数 M 的影响较小. 图 7 还表明, 当标签数量较少时, 适当减小 M 的取值, 可以提高 GJDB 算法的识别性能, 而对于标签数量较大的情况, 则可适当增大 M 的取值.

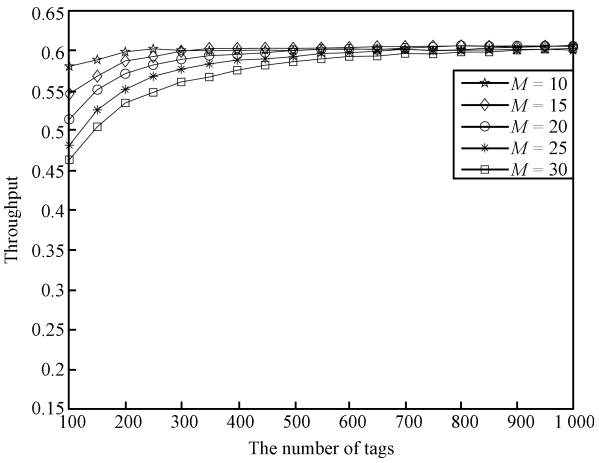

图 7 不同标签分组数目下吞吐率的变化

Fig. 7 Variation of throughputs under different grouping numbers of tags

图 8 为 $N = 48$、$M = 20$ 时, 4 种算法中阅读器的数据通信量 B_R 的比较. 从图 8 可以看出, GJDB 和 ASR 算法的 B_R 较小, 通信复杂度较低; GJDB 算法采取的标签分组策略显著降低了阅读器重复搜索的次数, 使得该算法的 B_R 略小于 ASR 算法. 由于在识别过程中阅读器需要向可识别标签发送 ID 请求数据位, 因此 GDF-BTA 算法的 B_R 明显大于其他三种算法.

图 9 比较了当 $N = 48$、$M = 20$ 时, 4 种算法中单个标签的平均数据通信量 B_T. 图 9 表明 GJDB 算法的 B_T 最小, 并且基本不受标签数量的影响, 其原因在于 GJDB 算法采取的分组策略在很大程度上减少了标签响应的次数, 同时标签采用动态方式来传送二进制代码, 这两者显著降低了标签的数据通信量. GDF-BTA 算法利用二叉树递归机制处理标签的碰撞, 因此, 其 B_T 小于 TSA 和 ASR 算法. 由于标签每次响应阅读器时都传送整个 ID, 使得 TSA

算法的 B_T 较大. 综合图 5、图 8 和图 9 可知, 与其他三种常用标签防碰撞算法相比, GJDB 算法的识别性能较好.

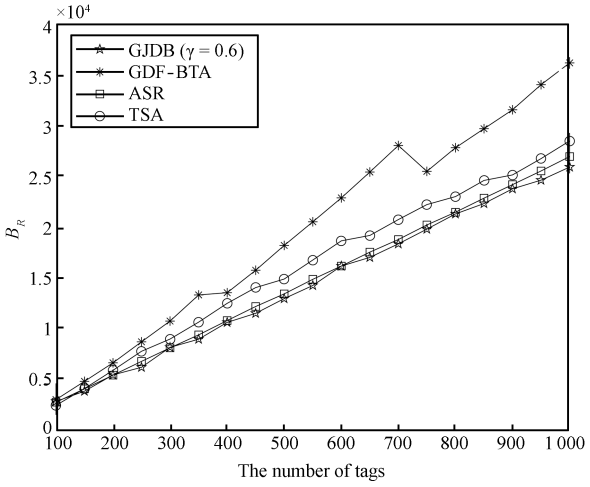

图 8　阅读器数据通信量 B_R

Fig. 8　Reader communication overhead B_R

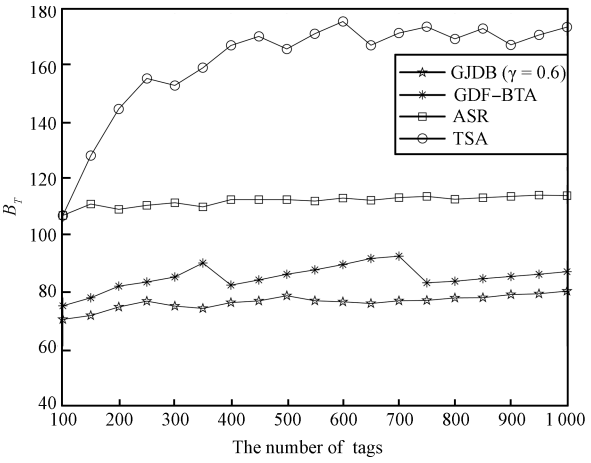

图 9　标签数据通信量 B_T

Fig. 9　Tag communication overhead B_T

4　结论

本文提出了一种基于分组机制的跳跃动态二进制标签防碰撞算法 GJDB, 并将其应用于解决 RFID 系统中的标签碰撞问题. 该算法通过依次对标签进行随机分组和最优分组, 大幅减少了每次响应阅读器查询的标签数目, 从而降低了标签发生碰撞的概率, 而对于碰撞标签, GJDB 算法则采用跳跃动态二进制算法进行识别. 理论分析和仿真结果表明, GJDB 算法显著提高了系统吞吐率, 降低了阅读器和标签的通信复杂度. 此外, 该算法对标签数目的估算精确度要求不高, 并且其识别效率不受标签 ID 长度的影响. GJDB 算法可广泛用于各种各样的应用场景中, 对大批量的物品识别和追踪管理具有非常重要的意义, 这必将进一步推动 "物联网" 的发展, 也为我国制定自己的超高频 RFID 系统标准提供了一定的理论参考依据.

References

1 Want R. An introduction to RFID technology. *IEEE Pervasive Computing*, 2006, **5**(1): 25−33

2 Vogt H. Multiple object identification with passive RFID tags. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Hammamet, Tunisia: IEEE, 2002. 6−11

3 RFID for Item Management — Air Interface, Part 3: Parameters for Air Interface Communications at 13.56 MHz, ISO Standard 18000-3, 2008

4 RFID for Item Management — Air Interface, Part 6: Parameters for Air Interface Communications at 860 MHz to 960 MHz, ISO Standard 18000-6, 2004

5 Cha J R, Kim J H. Dynamic framed slotted ALOHA algorithms using fast tag estimation method of RFID system. In: Proceedings of the 3rd IEEE Consumer Communications and Networking Conference. Piscataway, USA: IEEE, 2006. 768−772

6 Lee S R, Joo S D, Lee C W. An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification. In: Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services. San Diego, USA: IEEE, 2005. 166−172

7 Tian J H, Fan Y S, Zhu Y L, Zhang R Z. An improved EPC Gen-2 slot random anti-collision algorithm based on active strategy. In: Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems. Chengdu, China: IEEE, 2008. 685−688

8 Finkenzeller K. *RFID Handbook: Radio-Frequency Identification Fundamentals and Applications (Second Edition)*. England: John Wiley and Sons, 2003. 183−220

9 Shi X, Wei F, Huang Q, Wang L, Shi X W. Novel binary search algorithm of backtracking for RFID tag anti-collision. *Progress in Electromagnetics Research B*, 2008, **9**: 97−104

10 Luo Z W, Tan Z N, Ni Z C, Yen B J. Analysis of RFID adoption in China. In: Proceedings of the IEEE International Conference on E-Business Engineering. Hong Kong, China: IEEE, 2007. 315−318

11 Yu Song-Sen, Zhan Yi-Ju, Wang Zhi-Ping, Tang Zhong-Ping. Anti-collision algorithm based on jumping and dynamic searching and its analysis. *Computer Engineering*, 2005, **31**(9): 19−21
(余松森, 詹宜巨, 王志平, 唐忠平. 跳跃式动态树形反碰撞算法及其分析. 计算机工程, 2005, **31**(9): 19−21)

12 Law C, Lee K, Siu K Y. Efficient memoryless protocol for tag identification. In: Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications. Boston, USA: ACM, 2000. 75−84

13 Feng Bo, Li Jin-Tao, Zheng Wei-Min, Zhang Ping, Ding Zhen-Hua. A novel anti-collision algorithm of tag identification in RFID systems. *Acta Automatica Sinica*, 2008, **34**(6): 632−638
(冯波, 李锦涛, 郑为民, 张平, 丁振华. 一种新的 RFID 标签识别防冲突算法. 自动化学报, 2008, **34**(6): 632−638)

14 Kleinrock L, Lam S S. Packet switching in a multi-access broadcast channel: performance evaluation. *IEEE Transactions on Communications*, 1975, **23**(4): 410−423

15 Myung J, Lee W, Shih T K. An adaptive memoryless protocol for RFID tag collision arbitration. *IEEE Transactions on Multimedia*, 2006, **8**(5): 1096−1101

16 Bonuccelli M A, Lonetti F, Martelli F. Tree slotted Aloha: a new protocol for tag identification in RFID networks. In: Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks. New York, USA: IEEE, 2006. 603−608

17 Park J, Chung M Y, Lee T J. Identification of RFID tags in framed-slotted ALOHA with robust estimation and binary selection. *IEEE Communications Letters*, 2007, **11**(5): 452−454

18 Eom J B, Lee T J, Rietman R, Yener A. An efficient framed-slotted ALOHA algorithm with pilot frame and binary selection for anti-collision of RFID tags. *IEEE Communications Letters*, 2008, **12**(11): 861−863

19 Yang Jian, Zhan Yi-Ju, Wang Yong-Hua, Yu Song-Sen. An RFID anti-collision algorithm based on grouped dynamic frame and binary tree recursive identification. *Information and Control*, 2009, **38**(3): 257−263
(杨健, 詹宜巨, 王永华, 余松森. 一种基于分组动态帧与二叉树递归识别的射频识别防冲突算法. 信息与控制, 2009, **38**(3): 257−263)

20 Capetanakis J I. The Multiple Access Broadcast Channel: Protocol and Capacity Considerations [Ph. D. dissertation], Massachusetts Institute of Technology, USA, 1978

21 Identification Cards-Contactless Integrated Circuit(s) Cards-Proximity Cards-Part3: Initialization and Anti-Collision, ISO/IEC14443-3, 2001

王亚奇 南京邮电大学计算机学院博士研究生. 主要研究方向为 RFID 技术和复杂动态网络.
E-mail: wyq.njupt@gmail.com
(**WANG Ya-Qi** Ph. D. candidate at the College of Computer, Nanjing University of Posts and Telecommunications. His research interest covers RFID technology and complex dynamical networks.)

蒋国平 南京邮电大学自动化学院教授. 主要研究方向为混沌同步控制和复杂动态网络. 本文通信作者.
E-mail: jianggp@njupt.edu.cn
(**JIANG Guo-Ping** Professor at the College of Automation, Nanjing University of Posts and Telecommunications. His research interest covers chaos synchronization and control and complex dynamical networks. Corresponding author of this paper.)