RFID 中基于 EPC G1G2 的 DTJ-ALOHA 防碰撞算法仿真

杨晓娇[1] 叶 润[2] 吴必造[3]

[1](重庆交通大学信息技术中心 重庆 400074)

[2](电子科技大学 四川 成都 611731)

[3](中移物联网有限公司 重庆 401336)

摘 要 针对 EPC G1G2 标准中的动态帧时隙防碰撞算法,在标签数量较多时,会导致 Q 值频繁的更改,从而增加了阅读器的负担等问题,提出一种以碰撞、空闲和成功时隙的差值,作为判断 Q 值改变门限的 DTJ(Devalue Threshold Judgment)-ALOHA 算法。通过 Matlab 仿真与 EPC 中算法做对比,验证了 DTJ-ALOHA 算法无论在 Q 值改变次数、平均吞吐率,还是阅读器的开销上都优于常见改进算法。

关键词 动态帧时隙 ALOHA 算法 吞吐率 防碰撞 射频识别 RFID

中图分类号 TP312 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2017.06.054

SIMULATION OF DTJ-ALOHA ANTI-COLLISION ALGORITHM BASED ON EPC G1G2 IN RFID

Yang Xiaojiao[1] Ye Run[2] Wu Bizao[3]

[1](*Information Technology Centre, Chongqing Jiaotong University, Chongqing 400074, China*)

[2](*University of Electronic Science and Technology of China, Chengdu 611731, Sichuan, China*)

[3](*China Mobile IOT Company Limited, Chongqing 401336, China*)

Abstract In the dynamic frame slot ALOHA in the EPC G1G2 standard, when the number of tags is large, the Q value frequently changes, thereby increasing the burden on the reader and other problems. DTJ-ALOHA algorithm is proposed, which takes the difference of the collision, idle and success slots as the threshold to judge the Q value. Compared with the EPC algorithm through Matlab simulation, it is proved that the DTJ-ALOHA algorithm is better than the common improved algorithm in terms of the number of change of Q value, the average throughput, and the cost of the reader.

Keywords Dynamic frame slot ALOHA algorithm Throughput Anti-collision RFID

0 引 言

目前,国内外现有的针对 RFID 系统的大多数论文所研究以及改进的防碰撞算法,大都是基于 TDMA 的时分多路算法的基础上提出的[1-3]。根据算法思想的不同可分为下面几类:1) 基于分时隙思想的,识别结果不确定的 ALOHA 及其衍生算法;2) 基于二分查找思想的,识别结果确定的二进制及其衍生算法。ALOHA 虽然属于不确定性算法,但是由于其对硬件的要求没有二进制算法高而且其实现过程也相对简单且易于实现,因此 RFID 系统在阅读器作用域内存在大量标签待识别时(此时防碰撞算法需要满足:快速准确地识别阅读器作用域内的所有标签),由于系统实现简单,自效率高等优势首选就是 ALOHA 及其衍生算法[4]。因此,本文提出的 DTJ-ALOHA 算法也是基于 ALOHA 的基础上提出的。

现在国内外大多数针对基于 ALOHA 的改进算法的论文,主要是通过将阅读器作用域中存在的标签分成几组来对标签进行分流而达到减少碰撞概率的目的,或者通过某种策略来动态地改变帧长等方法来提高 RFID 系统的吞吐率。但这些算法大都停留在理论

收稿日期:2015-05-25。**杨晓娇**,工程师,主研领域:RFID 防碰撞算法。**叶润**,博士生。**吴必造**,工程师。

层面,没有和实际的国际标准协议中的防碰撞算法结合起来,因此,在工程中应用得并不广泛。本文就是针对 EPC G1G2 中推荐的 DSF(Dynamic Framed Slotted)-ALOHA 算法,对 EPC G1G2 协议中规定的 DFS-ALOHA 防碰撞算法在标签数量较多时,由于总时隙数的增加会导致空闲以及碰撞时隙也相应的增多,使得 Q 值发生频繁波动,以及频繁计算 Q_{fp}(Q 的浮点数形式)导致阅读器负担较重等缺陷,提出了一种根据碰撞、空闲以及成功时隙的差值作为 Q 值跳变门限的 DTJ-ALOHA 算法。最后通过在 Matlab7.0 平台上 Matlab 编程模拟算法的执行过程,对结果绘图对比分析,验证了 DTJ-ALOHA 算法不论在控制 Q 值改变次数、时隙吞吐率以及系统阅读器开销等性能上都优于推荐的 EPC G1G2 推荐的算法。

1 EPC 中的动态帧时隙 DFS-ALOHA 算法

EPC global 标准委员会的 EPC C1G2 中推荐的算法也是基于 ALOHA 的即 DFS-ALOHA 算法[5]。协议规定通过定义一个 Q 来根据时隙状态灵活改变帧长,阅读器通过向其作用域内广播发送 Query、QueryAdjust 和 QueryRep 等命令来改变帧长。并且根据标签的回复记录下当前时隙的状态,然后根据统计的标签状态来动态调整帧长,若 Q 值发生变化,则标签工作在 2^Q 的动态帧长下[6]。

1.1 EPC G1G2 中 DFS-ALOHA 的涉及到的命令简介

本节将对 EPC G1G2 协议中用到的命令(在第 2 节中 DTJ-ALOHA 算法中也会用到的命令)简介[8]。

Query (Q) 命令:目的是向标签发送 Q 值,标签从 Query (Q) 中获取 Q 值后选择 x 属于$[0,2^Q-1]$ 放到标签的 SC(Slot Count)中,当且仅当 SC 中存储的值通过 QueryRep (n) 命令自减为 0 时,标签才请求通信。

QueryAdjust (Q):协议中规定的更新时隙的命令,用于向标签广播用于计算时隙的参数 Q。标签从 QueryAdjust (Q) 命令中获取 Q 值后,先将其 SC 中数值清零,然后选择 x 属于$[0,2^Q-1]$ 放到标签的 SC (Slot Count)中。

QueryRep (n):减少 SC 值命令,该命令携带一个参数 n,阅读器作用域范围内标签收到该命令广播的 n 值后,根据 $SC = SC - n$ 计算 SC 的值将其重新存储在 SC 中(EPC G1G2 中 $n = 1$)。

1.2 EPC G1G2 中的防碰撞算法思想

EPC G1G2 中采用 DSF-ALOHA 算法识别标签的过程,如图 1 所示,主要是根据当前时隙的状态来动态地调整 Q 值从而改变帧长,调整方法是:1) 若为空闲时隙则 $Q_{fp} = Q_{fp} - C$;2) 若为碰撞时隙则 $Q_{fp} = Q_{fp} + C$;3) 若为成功时隙则 $Q_{fp} = Q_{fp}$。每个时隙过后就对 Q_{fp} 进行计算 Q = round(Q_{fp})(取整方法为四舍五入)。若本时隙结束后,经过计算发现 Q 不等于 Q_{start},即需要更新 Q 值,接着发送 QueryAdjust(Q) 命令对 Q 值进行更新;反之,则表示不需要更新 Q 值,只需阅读器继续处理下一个时隙即可,即发送 QueryRep(1)命令继续识别[9]。

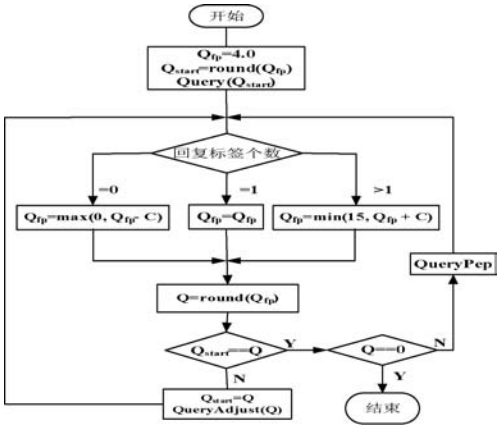

图 1 EPC G1G2 中建议调整 Q 值的方法(Q_{fp} 为 Q 的浮点形式)

在 EPC G1G2 协议规定 C 为浮点数,其取值范围为 0.1 ~ 0.5,且 C 取值由于 Q 的增大而减小[8]。这是因为当阅读器作用域内存在大量标签时,总时隙 N 也会很大,从而使得碰撞以及空闲时隙数也相应的增加,由于推荐算法在每寻呼一个时隙就要对 Q 进行一次浮点运算,当总时隙 N 很大时阅读器进行浮点数的运算量也很大,此时阅读器的负担就很大,即便取 $C = 0.1$ 也会同样会导致 Q 值频繁的更新。因此,针对上述不足,本文提出了一种改进的基于差值门限判决来决定 Q 值是否改变的 DTJ-ALOHA 算法。新算法不用进行浮点预算,并且再标签数增加时不会导致 Q 值的频繁更新。

2 DTJ-ALOHA 算法

在实际 RFID 系统中时隙数 N 不能为任意值,N 的取值只能是 2^Q($Q = 0,1,2,\cdots$) 则可以计算得出,标签根据接收到的 Q 值生成的时隙数可为 0,2,4,…,128,256,512,且当 Q 值越大对 Q 值进行加或者减 1 操作后,对 N 的波动会越大。本文根据碰撞、空闲、和成功时隙

个数的差值表达式来作为 Q 改变的依据,从而规避了上述问题,提出了一种新的算法。下面介绍 DTJ-ALOHA 算法是如何提出的。

在 ALOHA 算法的基础上面,可推导得出在 FS-ALOHA 算法中,某时隙中存在 k 个标签的概率为:

$$P(X=k)=\left(\frac{n}{k}\right)(p)^k(1-p)^{n-k}=\left(\frac{n}{k}\right)\left(\frac{1}{N}\right)^k\left(1-\frac{1}{N}\right)^{n-k} \quad (1)$$

同理,由式(1)令 $X=1$ 即可得出有且仅有一个标签选择当前时隙(表示当前时隙为成功时隙)的概率为:

$$P_{suc}=P(X=1)=\frac{n}{N}\left(1-\frac{1}{N}\right)^{n-1} \quad (2)$$

对式(2)进行 Matlab 仿真分别令 $N=16,32,64,128,256$ 时可以得出算法的吞吐率随标签个数的变化规律,如图 2 所示(从左到右依次为 $N=16,32,64,128,256$ 的图像)。

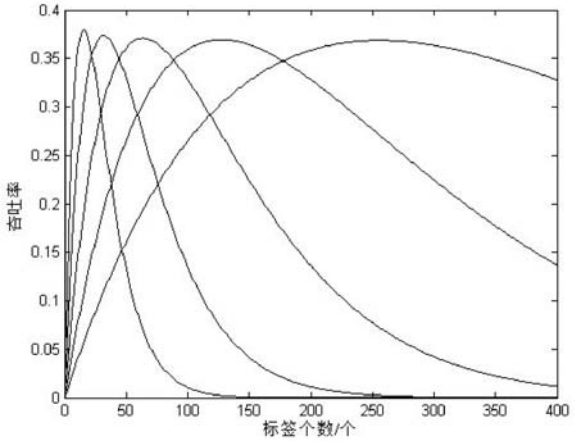

图 2　$N=16,32,64,128,256$ 时
FS-ALOHA 算法吞吐率与标签个数的关系

由图 2 可知,要保持吞吐率较高,则取两条曲线的交点间部分对应的时隙数(如标签个数在 ab 段时取时隙数 $N=64$),吞吐率较高的两条直线的交点即为时隙数为 N 的曲线与时隙数为 $2N$ 和 $N/2$ 的交点(以 $N=64$ 这条曲线的吞吐率为例分析,与 $2N$ 的交点即为 b 点与 $2/N$ 的交点即为 a 点)。可以推出时隙数分别为 N 以及 $2N$ 的两条吞吐率曲线的交点所对应的标签个数 $n_{N=2N}$ 为:

$$P_{SUC}(N=2N)=P_{SUC}(N=N)\Rightarrow n_{N=2N}=\frac{\lg 2}{\lg\left(\frac{2N-1}{2N-2}\right)}+$$

$$1\approx\frac{\lg 2}{\lg\left(\frac{2N-1}{2N-2}\right)}(n\geqslant 1) \quad (3)$$

类似地可求出时隙分别为 $N/2$ 以及 N 的两条吞吐率(成功识别时隙个数概率)曲线的交点标签数 $n_{N=\frac{N}{2}}$ 为:

$$P_{SUC}\left(N=\frac{N}{2}\right)=P_{SUC}(N=N)\Rightarrow n_{N=\frac{N}{2}}=\frac{\lg 2}{\lg\left(\frac{N-1}{N-2}\right)}+$$

$$1\approx\frac{\lg 2}{\lg\left(\frac{N-1}{N-2}\right)}(n\geqslant 1) \quad (4)$$

由式(1)令 $k=0$ 可以推出,没有一个标签选择某时隙的概率(该时隙为空闲时隙)P_{null} 为:

$$P_{null}=P(X=0)=\left(1-\frac{1}{N}\right)^n \quad (5)$$

同理 $k>1$ 可推出不止一个标签选择某时隙的概率(即该时隙为碰撞时隙)P_{coll} 为:

$$P_{coll}=P(k>1)=1-\frac{n}{N}\left(1-\frac{1}{N}\right)^{n-1}-\left(1-\frac{1}{N}\right)^n \quad (6)$$

以图 2 中 ab 段对应的 $N=64$ 的吞吐率曲线为例,该曲线的顶点(即吞吐率最高)所对应的 x 轴标签个数也大概为 64 左右,其余吞吐率曲线也是当 N 约等于 n 时,吞吐率最高。下面以 $N=16$ 时碰撞、成功以及空闲时隙占总时隙的概率,随标签个数的变化规律为例进行 Matlab 仿真,结果如图 3 所示。从图中的概率曲线可以得出当算法吞吐率较高时,碰撞概率与成功概率差值以及空闲概率与成功概率的差值较小。

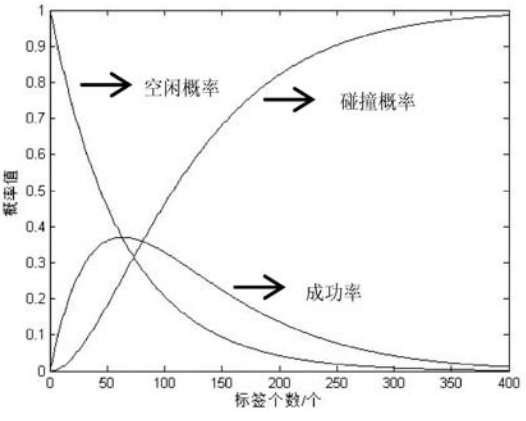

图 3　帧长为 $N=64$ 时 FS-ALOHA 算法时碰撞、成功以及空闲时隙概率随标签个数 n 的变化图

所以,本文提出了一种根据碰撞与成功以及空闲与成功时隙的差值作为阈值来控制 Q 值的改变,将 Q 值由 Q 变为 $Q+1$ 的阈值 Th^+ 定义为:$Th^+=N\times(P_{coll}-P_{suc})$,将 Q 值由 Q 变为 $Q-1$ 的阈值 Th^- 定义为:$Th^-=N\times(P_{null}-P_{suc})$ 将前面定义的式(5)、式(6)代入阈值 Th^+ 和 Th^- 表达式化简可得出:

$$Th^+=N\times(P_{coll}-P_{suc})=N\times\left[1-2\frac{n}{N}\left(1-\frac{1}{N}\right)^{n-1}-\left(1-\frac{1}{N}\right)^n\right] \quad (7)$$

$$Th^- = N \times (P_{null} - P_{suc}) = N \times \left[\left(1 - \frac{1}{N}\right)^n - \frac{n}{N}\left(1 - \frac{1}{N}\right)^{n-1}\right] \quad (8)$$

由式(7)和式(8)可以得出 Q 值及其与 Th^+ 和 Th^- 取值为表 1 所示。

表 1 Q 值与其对应的阈值 *Th*

Q 值	时隙数 N	Th^+ 值	Th^- 值
1	2	-1	2
2	4	0	1
3	8	0	2
4	16	0	3
5	32	1	5
6	64	3	10
7	128	7	20
8	256	14	40
9	512	29	79
10	1 024	58	157
⋮	⋮	⋮	⋮

表 1 中提前计算好了 Q 跳变的阈值,阅读器只需要根据每个时隙的状态做一个简单的比较以及加减法运算,这样就有效解决了本节开始提出的由于做浮点运算而导致阅读器负担增加的问题。

因此,下面简单介绍 DTJ-ALOHA 算法的程序流程图。

按照如图 4 所示的流程,当阅读器向标签发出通信请求后,根据回复的个数确定本时隙碰撞、空闲还是成功时隙。接着统计本帧中空闲、碰撞以及成功时隙的个数。

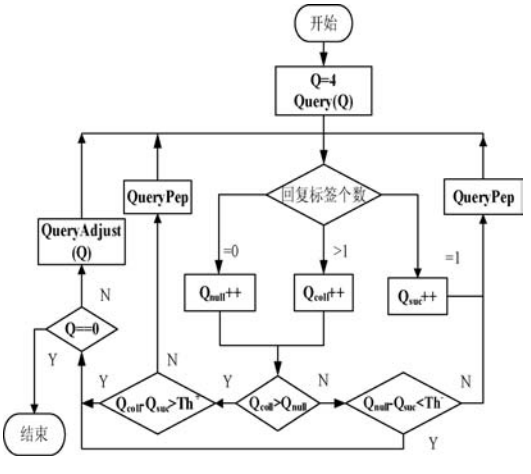

图 4 DTJ-ALOHA 算法的详细程序流程图

如果本时隙为非成功时隙,则比较碰撞时隙与空闲时隙大小,1)若 $Q_{coll} > Q_{suc}$ 将碰撞时隙与成功时隙差值与表 1 中对应 Th^+ 的阈值进行比较,如果统计的时隙数差超过阈值,则表示碰撞时隙过多,应增加时隙数,发送 QueryAdjust (Q) 命令调整 Q 值来更新时隙数,反之没有超过阈值表示时隙数不变发送 QueryRep (1)命令继续读取下一个时隙。2)若 $Q_{coll} \leqslant Q_{suc}$ 将空闲时隙与成功时隙差值与表 1 中对应 Th^- 的阈值进行比较,如果统计的时隙数之差超过阈值,则表示空闲时隙过多,应减少总时隙数,发送 QueryAdjust (Q) 命令通过调整 Q 值来更新总时隙数,反之没有超过阈值表示时隙数不变发送 QueryRep(1)命令继续读取下一个时隙。

由表 1 可以看出,DTJ-ALOHA 算法的阈值 Th^+ 和 Th^- 是根据 Q 值动态变化的,且 Q 值越大 Th^+ 和 Th^- 越大。这样就避免了在标签数量较大时,由于 Q 值的频繁改变而增加阅读器负担的问题。因此,DTJ-ALOHA 算法在标签数量不确定时,特别是个数动态变化,或者有大规模标签待识别时,优势明显。

3 DTJ-ALOHA 算法的仿真以及结果分析

本文的所有仿真实验都是在 Matlab7.0 平台上进行的。分别编程实现不同算法识别标签的过程,再统计运行的结果,为了减小偶然因素对算法评估造成影响,本文中所有图中的结果都是识别相同的标签 50 次后的结果所取的平均值。

首先分别编程实现 EPC G1G2 算法以及 DTJ-ALOHA 算法的执行过程,得出它们发送 QueryAdjust (Q) 改变 Q 值的次数对比如图 5 所示。

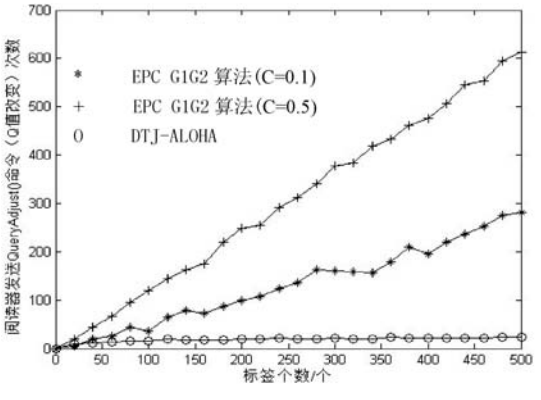

图 5 采用阈值跳变来更改 Q 值与 EPC G1G2 推荐方法改变 Q 值需要发送 QueryAdjust( ) 命令次数的对比图

由图 5 可看出 EPC G1G2 算法当标签个数增加时不论 C 取何知, Q 值改变次数都称线性增加,反之 DTJ-ALOHA 算法的控制在 30 以内,从而减少了阅读器的负担。

通过图 5 与表 2 可看出,不论标签多与少,改进算法都将 Q 值改变次数控制在一定范围 30 内。即改进算法相比于 EPC G1G2 算法适应性较好。

表 2　Q 值改变次数对比图

标签个数	EPC C = 0.5 Q 值改变次数	EPC C = 0.1 Q 值改变次数	DTJ-ALOHA Q 值改变次数	DTJ 相比于 C = 0.1 节省率
20	19	10	8	20%
60	68	23	14	39%
100	117	40	17	57%
200	234	100	19	81%
⋮	⋮	⋮	⋮	⋮
800	973	439	24	95%
1 000	1 233	590	22	99%

本文中将 RFID 系统中 ALOHA 算法的平均吞吐率定义为:有效时隙数(成功时隙数)/总时隙(阅读器识别完其作用域内全部标签共需要的时隙数)。

从图 6 可以看出,大规模标签识别过程中,改进算法 DTJ-ALOHA 的吞吐率稳定在 35.5% 左右,EPC G1G2 算法吞吐率稳定在 34.5%(C = 0.1)和 33%(C = 0.5)左右,验证了改进算法即 DTJ-ALOHA 阈值跳变算法比 EPC G1G2 算法的吞吐率更优。

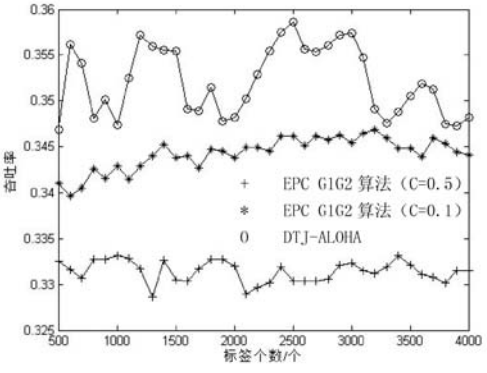

图 6　吞吐率比较图

当前针对动态帧时隙 ALOHA 的改进算法主要是基于对标签数估计算法,算法的主要思路是阅读器先读取完 N 个时隙的信息,并统计碰撞率空闲率,并根据某种算法来估计待识别标签的数量,然后调整 Q 值,这类常见动态 ALOHA 改进的算法,优势是 Q 值改变次数相对较小。但由于这类算法每次对待识别标签数进行估计前,都需要先统计 N 个时隙的状态,并以此为依据结合算法来调整帧长,因此这类改进算法适应性即吞吐率没有 DTJ-ALOHA 算法好。

由于阅读器开销大小直接决定了识别其作用范围内所有标签的耗时长短,也就直接影响了识别的准确率。是衡量算法效率的关键指标。为了证明本文提出的 DTJ-ALOHA 算法的可行性,下面将常见的动态帧时隙的改进算法与本文提出的 DTJ-ALOHA 算法在阅读开销上进行仿真对比。

根据表 3 中规定的命令长度,对 DTJ-ALOHA 算法、基于标签数估计的动态帧时隙 ALOHA 算法以及 EPC G1G2 算法对阅读器读取标签过程中的开销(发送数据量)做仿真。

表 3　EPC C1G2 协议命令/参数长度

命令/参数	功能	长度/bit
Query	22	22
QueryAdjust	9	9
QueryRep	4	4
ACK	18	18

图 6 和图 7 的结果已经验证了改进的 DTJ-ALOHA 算法在关键性能指标上不仅优于 EPC G1G2 推荐的算法也优于常见的改进(基于标签估计的动态帧时隙 ALOHA)算法。

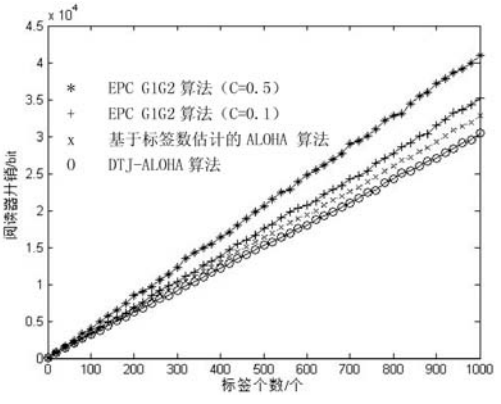

图 7　三种不同算法对应的阅读器开销对比图

4　结　语

本文在 EPC G1G2 的 DFS-ALOHA 算法基础上提出了一种基于差值门限判决的 DTJ-ALOHA 算法,改进算法采用差值判决替代 C 值来触发 Q 值的改变,因而不用进行浮点运算,减轻了阅读器的计算量。由于阈值的动态变化也使得算法在标签个数不确定时始终能保持较优的吞吐率。相比于其他改进的动态帧时隙算法(读完 N 个时隙再改变 Q 值)本算法延续了 EPC G1G2 快速动态调整 Q 值的优势。然后通过在 Matlab7.0 平台上编程模拟算法的执行过程,仿真证明了 DTJ-ALOHA 算法将 Q 值改变次数控制在 30 以内,且算法的平均吞吐率稳定在 35.5% 左右。最后通过仿真证明了本文提出的 DTJ-ALOHA 算法在阅读器开销这一衡量算法的关键指标上是优于常见的基于标签估计的改进的动态帧时隙 ALOHA 算法,且 DTJ-ALOHA 算法并没有额外的增加寻呼命令,因此可以在 EPCG1G2 协议上直接实现。由于基于 EPC G1G2 协议的实现的 RFID 系统常用于有大规模标签待识别的系统比如大型港口、大型商场等实际应用环境,因此本

文提出的 DTJ-ALOHA 算法有一定的实用价值,可应用在 RFID 系统大规模标签待识别环境中。

参 考 文 献

[1] 孙其博,刘杰,黎羴,等. 物联网:概念、架构与关键技术研究综述[J]. 北京邮电大学学报,2010,33(3):2-3.

[2] 黄玉兰,夏璞,夏岩,等. 物联网射频识别(RFID)核心技术详解[M]. 北京:人民邮电出版社,2012:22-25.

[3] 石封查,崔琛,余剑. 基于标签运动的一种新型 RFID 防碰撞算法[J]. 计算机科学,2013,40(6):76-79.

[4] Finken Zeller K. 射频识别(RFID)技术—无线电感应的应答器和非接触 IC 卡原理与应用[M]. 2 版. 北京:电子工业出版社,2001:98-101.

[5] EPC Global. EPC Radio-Frequency Identity Protocols Gengernation-1[S]. American:UCC,2013-10-31.

[6] 肖海慧,王红明. 一种基于 DFSA 防碰撞协议的 FBF 改进算法研究[J]. 计算机应用与软件,2013,30(7):305-308.

[7] Cha J R, Kim J H. Dynamic framed slotted ALOHA algorithms using fast tag estimation method for RFID system [C]//Consumer Communications and NETWORKING Conference,2006. Ccnc. 2006:768-772.

[8] EPC Global. EPC Radio-Frequency Identity Protocols Gengernation-2 UHF RFID Protocol for Communication at 860MHZ ~ 960MHZ Version[S]. California: EPC Global,2008.

[9] 徐圆圆,曾隽芳,刘禹. 基于 Aloha 算法的帧长及分组数改进研究[J]. 计算机工程,2008,3(12):57-59.

[10] 吴淼,刘德盟,张钊锋. 基于 EPC Gen2 防碰撞机制的研究与优化[J]. 微电子学与计算机,2013,30(5):101-104.

(上接第 235 页)

[7] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation [C]//Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. New York, NY, USA: ACM Press, 2000: 1-12.

[8] Han J, Pei J, Mortazavi-Asl B, et al. FreeSpan: frequent pattern-projected sequential pattern mining[C]//Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM Press, 2000: 355-359.

[9] Pei J, Han J, Mortazavi-Asl B, et al. PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth [C]//Proceedings of the 17th International Conference on Data Engineering. IEEE Computer Society, 2001: 215-224.

[10] 张坤,朱扬勇. 无重复投影数据库扫描的序列模式挖掘算法[J]. 计算机研究与发展,2007,44(1):126-132.

[11] 张利军,李战怀,王淼. 基于位置信息的序列模式挖掘算法[J]. 计算机应用研究,2009,26(2):529-531.

[12] 刘栋,尉永清,薛文娟. 基于 Map Reduce 的序列模式挖掘算法[J]. 计算机工程,2012,38(15):43-45.

[13] 吴信东,谢飞,黄咏明,等. 带通配符和 One-Off 条件的序列模式挖掘[J]. 软件学报,2013,24(8):1804-1815.

[14] 刘端阳,冯建,李晓粉. 一种基于逻辑的频繁序列模式挖掘算法[J]. 计算机科学,2015,42(5):260-264.

[15] 朱扬勇,熊赟. DNA 序列数据挖掘技术[J]. 软件学报,2007,18(11):2766-2781.

[16] 熊赟,陈越,朱扬勇. DnaReSM:一个基于多支持度的 DNA 重复序列挖掘算法[J]. 计算机科学,2007,34(2):211-212,封四.

[17] 周溜溜,业宁,徐昇,等. 基于频繁子树挖掘的 DNA 重复序列识别方法[J]. 微电子学与计算机,2011,28(9):193-196,201.

[18] 姜华,孟志青,周克江. DNA 序列频繁近似模式挖掘[J]. 生物信息学,2013,11(1):11-15.

[19] 毛国君,杨静欣. 一种基于关联矩阵的高效 DNA 序列挖掘算法[J]. 计算机科学与应用,2015,5(8):271-277.

(上接第 266 页)

[8] Yuan X, Dai X, Zhao J, et al. On a novel multi-swarm fruit fly optimization algorithm and its application[J]. Applied Mathematics & Computation, 2014, 233(3): 260-271.

[9] Pan W T. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example[J]. Knowledge-Based Systems, 2012, 26(2): 69-74.

[10] 郑晓龙,王凌,王圣尧. 求解置换流水线调度问题的混合离散果蝇算法[J]. 控制理论与应用,2014,31(2):159-164.

[11] 杨书,舒勤,何川. 改进的果蝇算法及其在 PPI 网络中的应用[J]. 计算机应用与软件,2014,31(12):291-294.

[12] Shan D, Cao G H, Dong H J. LGMS-FOA: An Improved Fruit Fly Optimization Algorithm for Solving Optimization Problems [J]. Mathematical Problems in Engineering, 2013, 2013(7): 1256-1271.

[13] Lin Y C, Hwang K S, Wang F S. A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems[J]. Computers & Mathematics with Applications, 2004, 47(8-9): 1295-1307.

[14] Deep K, Singh K P, Kansal M L, et al. A real coded genetic algorithm for solving integer and mixed integer optimization problems[J]. Applied Mathematics & Computation, 2009, 212(2): 505-518.

[15] 唐俊. PSO 算法原理及应用[J]. 计算机技术与发展,2010,20(2):213-216.

[16] 梅觅,薛惠锋,谷雨. 旅行商问题的改进差分进化方法[J]. 信息技术,2011(2):20-23.