周逸, 董自健, 李兆鹏. 基于动态初始帧长调整的 DFSA 防碰撞方法研究[J]. 智能计算机与应用,2026,16(8):115-120.
DOI:10.20169/j.issn.2095-2163.24122303

基于动态初始帧长调整的 DFSA 防碰撞方法研究

周 逸, 董自健, 李兆鹏
(江苏海洋大学 电子工程学院, 江苏 连云港 222005)

摘 要: 随着 RFID 技术的广泛应用,尤其在物流、智能家居等领域的快速发展,如何提高 RFID 系统的效率,成为了研究的热点问题。基于 DFSA(动态帧时隙 ALOHA)协议的防碰撞方法,通过动态调整帧时隙数量,有效降低了系统碰撞率并提升了吞吐量。然而,大多数现有的 DFSA 算法假设标签总量固定不变,而在实际应用中,标签数量往往随着环境变化而波动。针对这一问题,本文提出了一种改进的 DFSA 防碰撞方法,根据阅读器单次访问中采集到的成功时隙数,动态调整初始帧长,适配标签数量的动态变化的场景。本文采用真实的库存数据集进行仿真实验,仿真结果表明,该方法在动态标签数量场景下可有效提升系统吞吐率,减少时隙消耗,从而提升整体传输效率。

关键词: RFID; DFSA; 初始帧长调整; 仿真实验; 吞吐率
中图分类号: TP391.44 **文献标志码**: A **文章编号**: 2095-2163(2026)08-0115-06

Research on DFSA collision avoidance method based on dynamic initial frame length adjustment

ZHOU Yi, DONG Zijian, LI Zhaopeng
(School of Electronic Engineering, Jiangsu Ocean University, Lianyungang 222005, Jiangsu, China)

Abstract: With the widespread application of RFID technology, especially the rapid development in fields such as logistics and smart homes, improving the efficiency of RFID systems has become a hot research topic. The anti-collision method based on the DFSA (Dynamic Frame Slotted ALOHA) protocol effectively reduces system collision rates and improves throughput by dynamically adjusting the number of frame slots. However, most existing DFSA algorithms assume a fixed total number of tags, whereas in practical applications, the number of tags often fluctuates with changes in the environment. To address this issue, this paper proposes an improved DFSA anti-collision method that dynamically adjusts the initial frame length based on the number of successful slots collected by the reader during a single access, in order to adapt to dynamic tag quantity scenarios. simulation experiments are cartried out based on real-world inventory datasets and demonstrate that the proposed method can significantly improve system throughput, reduce the number of slots consumed, and enhance overall transmission efficiency in dynamic tag quantity scenarios.

Key words: RFID; DFSA; initial frame length adjustment; simulation experiment; throughput

0 引 言

射频识别技术(RFID)是一种利用无线射频信号进行数据传输和识别的非接触式自动识别技术,已在物流管理、医疗保健等多个领域得到广泛应用[1]。与传统条形码技术相比,RFID 具有更高的便捷性和安全性,能够实现更快速、更精确的识别[2]。然而,随着标签数量的增多,RFID 系统在读取多个标签过程中常出现标签响应冲突问题,导致系统工作效率下降[3-4]。为解决这一问题,防碰撞技术显得尤为重要,其中 ALOHA 算法因其实现简单、效率较高,在实际应用中得到了广泛使用[5]。

ALOHA 协议是一种早期的无线通信协议,最初用于解决多用户在共享信道中随机发送数据时的冲突问题[6]。其具有简单易实现的特点,但由于用户传输是完全随机的,冲突概率较高,导致信道利用率

作者简介: 周 逸(2000—),男,硕士研究生。主要研究方向:物联网通信技术; 李兆鹏(2001—),男,硕士研究生。主要研究方向:计算机视觉应用。
通信作者: 董自健(1973—),男,博士,教授。主要研究方向:检测与控制,工业自动化。Email:dzjian@126.com。

收稿日期: 2024-12-23

较低(纯 ALOHA 理论最大利用率为 18.4%,时隙 ALOHA 为 36.8%)[7-8]。为解决这一问题,ALOHA 的多个改进版本相继提出,其中包括时隙 ALOHA(Slotted ALOHA):通过将时间划分为固定长度的时隙以降低碰撞发生的概率[9];帧时隙 ALOHA(Framed Slotted ALOHA):将时间进一步划分为帧,并在帧内进行时隙划分,提高系统在复杂环境下的工作效率[10]。随着多用户通信需求的增加,固定时隙方案在灵活性和资源利用率上逐渐暴露出不足。

动态帧时隙 ALOHA(Dynamic Framed Slotted ALOHA)是在帧时隙 ALOHA 的基础上引入帧长度的动态调整机制,以适应不同网络条件和用户需求[11]。动态帧时隙 ALOHA 可以根据当前信道状态和剩余待传输数据的数量,实时优化帧长度,减少冲突和空闲时隙,提升系统的吞吐量和资源利用率。该协议在 RFID 系统、物联网设备通信以及其他需要高效数据传输的场景中表现出显著优势,成为多用户共享信道通信中的重要技术[12]。DFSA 通过捕捉时隙信息,动态地调整剩余标签数量所在帧的帧长,使算法在高速,高吞吐量的场景下具备更高的系统效率[13]。明东岳等[14]提出了一种基于 BP 神经网络的剩余标签数量预测方法,通过离线训练的方式提升工作时的系统效率。杨超等[15]结合 DFSA 和 LSTM 深度神经网络,对剩余标签数量进行准确预测并调整帧长,减少了时隙消耗;陈祥营等[16]通过伪随机数生成标签特征值与分组因子进行应答的方法,缩短识别时间。

现有研究大多数是针对标签总量不变的场景,在各种改进的 DFSA 算法中,初始帧长大多为预先设定值。在实际应用中,标签的总量不是固定不变的,例如物流与供应链管理系统中,仓库内部的货物流动会影响标签的总量;在智能家居系统中,购买的 RFID 新设备也会影响标签的总量。本文在 DFSA 算法的基础上,提出根据阅读器单次访问过程中采集得到的全部成功时隙数来确定初始标签数量,动态的调整初始帧长,使系统在标签总量动态变化的场景下有着更高的灵活性与系统效率。

1 ALOAH 协议及其演变

1.1 ALOHA

ALOHA 协议是一种简单的随机接入协议,该协议允许多用户通过共享信道传输数据。ALOHA 的核心思想是让终端设备随时尝试发送数据,如果发生数据碰撞,则重新尝试。ALOHA 工作原理如图 1 所示。

标签 1 独立完成数据包发送,阅读器数据接收成功;标签 2 和标签 3 在发送数据时发生部分碰撞的情况,阅读器数据接收失败,两个标签经过随机退避时延后重新尝试发送数据包。

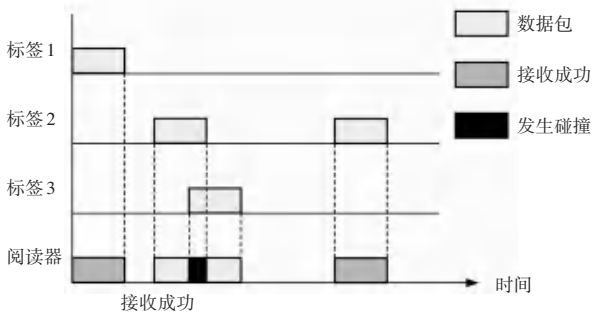


图 1 ALOHA 协议工作原理

Fig. 1 Working principle of ALOHA protocol

假设单位时间内平均发送的数据包数为 G、吞吐率为 S、数据包传输的时间为 T_{frame}、时间间隔为 T_{slot}、数据包发生碰撞的概率为 P_{coll}、数据包发生成功的概率为 P_{succ}。各标签发送都是独立的且发送时间服从泊松分布,那么在一个单位时间里,系统发生碰撞的概率为:

$$P_{\text{coll}} = 1 - e^{-2G} \tag{1}$$

系统发送成功的概率为:

$$P_{\text{succ}} = 1 - P_{\text{coll}} = e^{-2G} \tag{2}$$

吞吐率为单位时间内成功传输的数据包数量,可表示为:

$$S = G \cdot P_{\text{succ}} = G \cdot e^{-2G} \tag{3}$$

对式(3)求导并令其为 0,可以计算得到最大吞吐率为 0.183。

1.2 时隙 ALOHA

时隙 ALOHA 是对 ALOHA 协议的一种改进,将时间划分为固定长度的时隙,标签只能在时隙开始时进行数据发送,使数据包的发送具有同步性。时隙 ALOHA 工作原理如图 2 所示。

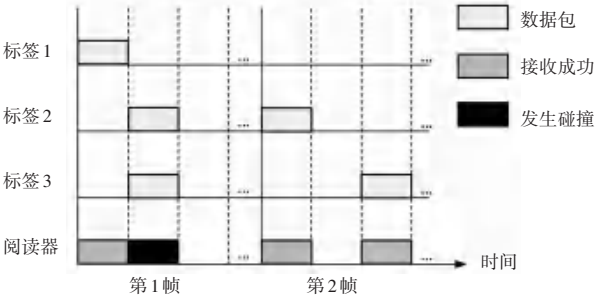


图 2 时隙 ALOHA 协议工作原理

Fig. 2 Working principle of Slotted ALOHA protocol

时隙 ALOHA 的工作过程分为 4 个部分：

(1)时间同步：整个系统时间同步，时间被划分为固定长度的时隙，每个时隙长度略大于一个数据包的传输时间。

(2)数据包发送：标签在时隙刚开始时发送数据包。

(3)碰撞检测：如果在一个时隙中有多个数据包传输，则发生完全碰撞。

(4)数据重传：当数据包发生碰撞时，碰撞标签会重新随机(通常是基于指数退避算法)选择时隙发送数据。

与 ALOHA 不同，时隙 ALOHA 通过同步时隙减少了碰撞的概率，由于时隙 ALOHA 只会发生完全碰撞，因此时隙 ALOHA 的吞吐率公式为：

$$S_{SA} = G \cdot e^{-G} \tag{4}$$

对公式(4)求导并令其为 0，可以计算得到最大吞吐率为 0.368，比 ALOHA(0.183)要高得多。

1.3　帧时隙 ALOHA

帧时隙 ALOHA 的核心思想是将时间划分为固定长度的帧，每个帧又划分为固定数量的时隙。标签在帧内随机选择时隙发送数据包，发生碰撞后只能在下一帧进行重新选择，以此类推。帧时隙 ALOHA 工作原理如图 3 所示。

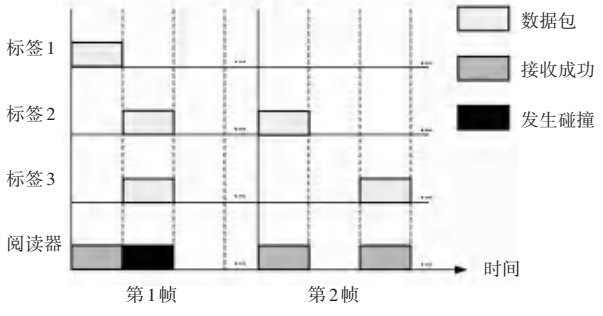


图 3　帧时隙 ALOHA 协议工作原理

Fig. 3　Working principle of framed slotted ALOHA protocol

标签在帧内随机选择时隙进行数据包发送，再由阅读器进行时隙判断。阅读器遍历帧内的每一个时隙，若该时隙有且仅有一个标签发送数据包，则该时隙为成功时隙，阅读器成功收集到该标签的数据；如果该时隙有多个标签发送数据包，则该时隙为碰撞时隙，碰撞的标签根据内部算法重新选择时隙，并在下一帧开始时重新尝试发送；如果该时隙没有标签发送数据包，则该时隙为空闲时隙。

帧结构的引入提升了协议的灵活性，但也增大了协议实现复杂度。面对不同场景，如何合理划分帧和时隙，是帧时隙 ALOHA 协议面临的挑战。

1.4　动态帧时隙 ALOHA

动态帧时隙 ALOHA 是帧时隙 ALOHA 的改进版本，该协议里的帧长和时隙划分是动态的，设备根据网络负载情况合理地调整帧长和时隙数量，该机制具备极强的灵活性，能够最大限度地减少碰撞，提高网络吞吐率，但算法实现起来也最为复杂。动态帧时隙 ALOHA 工作原理如图 4 所示。

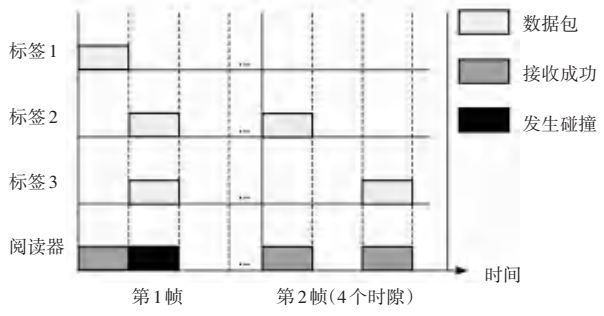


图 4　动态帧时隙 ALOHA

Fig. 4　Dynamic Frame Slotted ALOHA

标签在初始划分的帧长和时隙数内随机选择时隙进行数据包发送，当标签 2 和标签 3 在第 1 帧内发生信号碰撞时，会在第 2 帧重新发送。在第 1 帧结束时，阅读器会根据第 1 帧的成功时隙数，碰撞时隙数和空闲时隙数，预测下 1 帧要发送的剩余标签数量，动态的调整第 2 帧的帧长和时隙数设置。图 4 相比于图 3，减少了第 2 帧中冗余时隙带来的时间开销。

由于计算机的硬件特性，时隙分配选择 2 的幂次方能确保算法更加高效，且在硬件上易于实现，能够更好地匹配时间同步机制，确保时隙对齐与协同工作的效率。在许多无线协议和标准中(例如 IEEE 802.15.4、LoRa 等)，256 时隙被证实是兼顾效率与工程可行性的上限，动态帧时隙 ALOHA 协议里的帧长和时隙数设置见表 1[17]。

表 1　动态帧时隙 ALOHA 帧长设置与标签数量的关系

Table 1　Relation ship between Dynamic Frame Slotted ALOHA frame length setting and the number of tags

帧长	4	8	16	32	64	128	256
标签数量	1~5	6~11	12~22	23~45	46~88	89~177	177 以上

2 动态初始帧长调整的 DFSA 算法设计

2.1 DFSA 常见标签预测方法

根据 DFSA 算法的原理可知,预测下一帧的剩余标签数量是算法实现的关键。常见的标签预测方法有 LowBound 估计法、Schoute 估计法、Vogt 估计法和 Q 值预测法等[18]。

LowBound 估计法是一种基于最小剩余时隙的预测方法。假设上一帧检测的碰撞时隙数为 S_{coll},那么基于 LowBound 估计法预测的下一帧剩余标签数量 N_{LB} 为:

$$N_{\text{LB}} = 2 \cdot S_{\text{coll}} \tag{5}$$

这种估计法适合网络负载较低的场景,随着标签数量的增多,两个以上标签碰撞的碰撞时隙数量也会增多,预估准确率会大幅下降。

Schoute 估计法是一种基于动态负载估算的标签预测方法,结合系统负载信息,分析历史成功和碰撞时隙数据,计算每个时隙的剩余标签。根据理想情况计算,一个碰撞时隙的平均标签数量为 2.39。假设上一帧检测的碰撞时隙数为 S_{coll},那么基于 Schoute 估计法预测的下一帧剩余标签数量 N_S 为:

$$N_S = \lceil 2.39 \cdot S_{\text{coll}} \rceil \tag{6}$$

相较于 LowBound 估计法,该算法的预测准确率有一定提升,但随着标签数量的增加,标签碰撞情况变得更加复杂,估计准确率会有所下降。

Vogt 估计法是一种通过时隙情况来进行剩余标签预测的算法。假设上一帧包含成功时隙数、碰撞时隙数和空闲时隙数为 N_1, N_c, N_0,结合不等式可得:

$$\varepsilon = \min \left| \begin{pmatrix} N_1 \\ N_0 \\ N_c \end{pmatrix} - \begin{pmatrix} S_1 \\ S_0 \\ S_c \end{pmatrix} \right| \tag{7}$$

通过计算 ε 的最小值来计算剩余标签数量。该算法适合负载较高或时隙冲突较高的情况,可有效减少冲突并提高系统吞吐量。

Q 值预测法通常是基于强化学习中的 Q 值学习算法,通过与环境交互,学习每个时隙选择的期望回报[19]。在动态帧时隙 ALOHA 中,每个标签根据 Q 值调整自己的传输策略。如果某个时隙的 Q 值较高,节点更倾向于选择该时隙进行数据传输,从而减少冲突。该算法灵活性高,能有效减少碰撞,但计算难度较大,时耗较高[20]。

基于 BP 神经网络的剩余标签数量预测方法是通过搭建特定结构的 BP 神经网络,建立上一帧读取情况与剩余标签数量之间的映射关系,实现对标签数量的预估[14]。该方法通过模拟数据集预先完成 BP 神经网络模型训练,实际应用中只需要输入上一帧的成功时隙数,碰撞时隙数和空闲时隙数到训练好的模型,进行一次简单的前向计算即可得出预测值。算法在提高系统吞吐率的同时,计算所需的时耗也小。

Vogt 估计法、Q 值预测法、BP 神经网络预测法 3 种方法的系统吞吐率对比如图 5 所示。

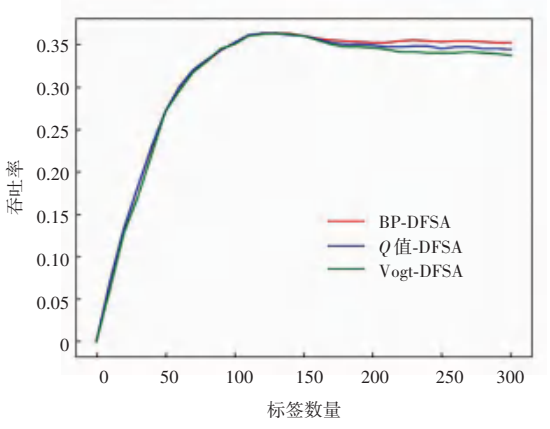


图 5 三种 DFSA 标签预测算法的吞吐率对比

Fig. 5 Through put comparison of three DFSA tag prediction algorithms

如图 5 所示,基于 BP 神经网络的剩余标签预估算法在处理标签数量较多的场景时,综合性能最优。本文选择使用 BP 神经网络作为改进算法中的标签预测算法部分。

2.2 初始帧长调整的 DFSA 改进算法

DFSA 算法虽然解决了阅读器在一次查询过程中时隙资源浪费的情况,但在初始标签总量缓慢变化的场景下,由于 DFSA 的初始帧长是固定的,标签总量增大会导致第 1 帧的碰撞时隙增多,标签总量减少会导致第 1 帧的空闲时隙增多,两种情况均会影响系统的整体效率。

由于阅读器在时隙查询过程中,发送成功的标签会停止阅读器响应,发生数据包碰撞的标签会在下一帧继续尝试发送,阅读器直到在某一帧中检测的成功时隙数和碰撞时隙数都为 0,视为所有标签全都发送完毕。阅读器在查询过程中所有帧内的成功时隙数之和等于初始标签总量。根据本次计算得到的标签总量近似作为下一次查询标签总量,对初始帧长进行调整。动态初始帧长调整的 DFSA 算法原理如图 6 所示,算法流程如图 7 所示。

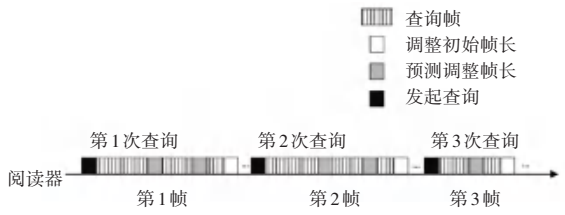


图 6　动态初始帧长调整的 DFSA 改进算法原理图

Fig. 6　Principle diagram of the improved DFSA algorithm with dynamic initial frame length adjustment

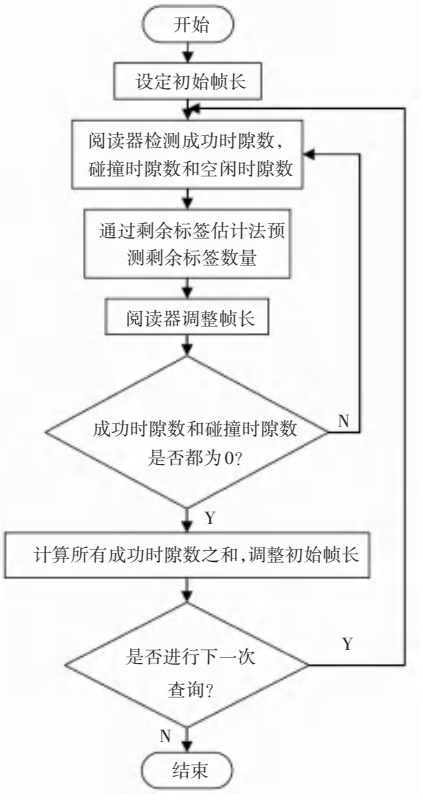


图 7　动态初始帧长调整的 DFSA 改进算法流程

Fig. 7　Flowchart of the improved DFSA algorithm with dynamic initial frame length adjustment

3　仿真和结果分析

3.1　仿真设计

本次仿真环境选用 PyCharm 平台，仿真步骤如下：

(1)采用江苏省盐城市射阳县澜儿换鞋吧每日的鞋子库存数量来模拟标签总量的变化过程。

(2)根据曲线提供的标签数量，分别开展传统 DFSA 算法和改进算法的模拟实验，传统算法初始帧长设置为 128。由于数据集最低鞋子数量为 50，改进算法的初始帧长设置了 64、128、256 这 3 种帧长选择(标准参考表 1)。标签预估算法采用基于 BP 神经网络剩余标签预估法。

(3)计算标签数量对应的吞吐率，得出吞吐率比较图。

2025 年 1 月 1 日至 3 月 15 日鞋子库存数量如图 8 所示。

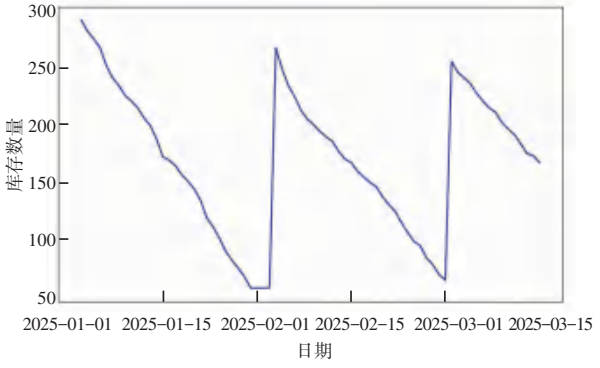


图 8　鞋子库存数量曲线图

Fig. 8　Shoe inventory quantity curve

3.2　结果分析

在动态帧时隙 ALOHA 协议中，吞吐率可表示为成功时隙数与总时隙数的比值。初始帧长动态调整的 DFSA 算法与传统的初始帧长固定的 DFSA 算法吞吐率对比如图 9 所示。

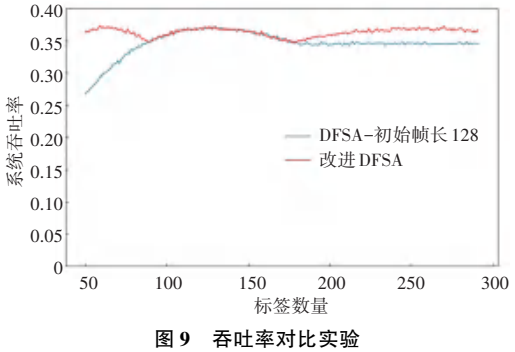


图 9　吞吐率对比实验

Fig. 9　Throughput comparison experiment

加入了动态初始帧长调整策略的改进 DFSA 算法能够在大部分标签数量环境中保持较高的系统吞吐率，整体上吞吐率要高于传统 DFSA 算法。

4　结束语

本文针对现有的动态帧时隙 ALOHA 算法依赖相对不变的标签数量环境存在的问题，提出了通过阅读器单次访问中所有帧内的成功时隙数之和来计算初始帧长，并作为下一次初始帧长调整的依据。该算法提高了整体系统的吞吐率，减少了时隙消耗，原理简单，使动态帧时隙 ALOHA 可以更灵活地应用在标签数量变化复杂的环境中。

参考文献

[1] 高洁，黄梅. 基于 RFID 的造纸行业仓库系统设计[J]. 造纸科

学与技术, 2024, 43(4): 109-112.
[2] 丁治国. RFID 关键技术研究与实现[D]. 合肥: 中国科学技术大学, 2009.
[3] 邓以回. 密集阅读器环境下 RFID 防碰撞算法研究[D]. 厦门: 厦门大学, 2022.
[4] 周伟辉. RFID 系统多标签自适应识别防碰撞算法研究与性能分析[D]. 赣州: 江西理工大学, 2017.
[5] 许若钰. RFID 多标签防碰撞识别技术研究[D]. 南京:南京信息工程大学, 2021.
[6] 单剑锋, 陈明, 谢建兵. 基于 ALOHA 算法的 RFID 防碰撞技术研究[J]. 南京邮电大学学报(自然科学版), 2013, 33(1): 56-61.
[7]王冬云, 张维平, 汪志佳. 基于改进 ALOHA 算法的 RFID 标签防碰撞研究[J]. 包装工程, 2021, 42(17): 244-248.
[8] 金浩, 王默, 李雁, 等. 基于 ALOHA 软件的安全设计优化分析[J]. 化肥设计, 2023, 61(4): 23-26.
[9] 侯培国, 王志轩, 严晨. 基于 RFID 标签的防碰撞算法改进[J]. 计算机科学, 2019, 46(S2): 359-362.
[10]阙大顺, 汪盛虎, 张浩. 帧时隙 ALOHA 的快速防冲突算法[J]. 计算机工程与应用, 2009, 45(35): 49-51.
[11]孙文胜, 金陈敏. 新型的 RFID 动态帧时隙 ALOHA 防碰撞算法[J]. 信息与控制, 2012, 41(2): 233-237.
[12]王达, 李晓武. 动态 RFID 系统中一种准确标签估计的动态帧时隙 ALOHA 算法[J]. 铁道学报, 2018, 40(7): 88-92.
[13]雷隆毓, 蒋荣斌, 陈子妍, 等. 基于标签分组的动态帧时隙 ALOHA 算法[J]. 计算机工程与设计, 2021, 42(4): 908-913.
[14]明东岳, 王尚鹏, 雷鸣,等. 结合 BP 神经网络的动态帧时隙 ALOHA 改进算法[J]. 小型微型计算机系统, 2021, 42(9): 1920-1923.
[15]杨超, 赵一帆, 李波, 等. 基于深度学习的 RFID 标签防碰撞算法研究[J]. 现代电子技术, 2021, 44(22): 21-25.
[16]陈祥营, 朱正伟. 基于 RFID 系统的防碰撞算法研究与改进[J]. 电子测量技术, 2021, 44(11): 84-89.
[17]周少珂, 邓淼磊. ALOHA 标签防碰撞算法综述[J]. 计算机工程与应用, 2017, 53(14): 9-17.
[18]赵跃. 煤矿安全预警系统中 RFID 防碰撞算法的研究[D]. 唐山:河北联合大学, 2014.
[19]杨子薇. 基于 *Q* 值的 RFID 防碰撞算法研究[D]. 南宁:广西大学, 2018.
[20]宋梓旭, 蒋毅, 张若南, 等. 一种优化 Q 值调整的 RFID 标签防碰撞算法[J]. 计算机工程, 2017, 43(9): 81-86.