applied sciences

MDPI

Article

Improved Anti-Collision Algorithm for the Application on Intelligent Warehouse

Yishu Qiu, Yezi Xu, Lvqing Yang *, Jinsheng Lu and Dingzhao Li

Software College, Xiamen University, Xiamen 361005, China; ysqiu.mail@gmail.com (Y.Q.); xuyezi1995@yeah.net (Y.X.); lujinshenga3000@163.com (J.L.); 24320181153597@stu.xmu.edu.cn (D.L.)

* Correspondence: lqyang@xmu.edu.cn; Tel.: +86-1865-000-5101

Received: 29 December 2018; Accepted: 10 April 2019; Published: 17 April 2019

Abstract: As an important part of economic development, warehousing logistics also needs to be transformed and upgraded in order to adapt to the development of the new situation. The RFID reader records the related information of the goods to improve the efficiency of warehouse operation by identifying the RFID tags attached to the goods in batches. This paper also proposes an improved group-based anti-collision algorithm (GMQT) to solve the problem of tag collision in the process of Radio Frequency Identification (RFID) identification. The simulation results show that the GMQT algorithm improves the recognition efficiency of the system. The algorithm has the advantages of small data transmission and stable performance; in particular, the recognition efficiency is not affected by the number of tags.

Keywords: RFID; intelligent warehouse; IoT; anti-collision algorithm

1. Introduction

With the rapid development of logistics industry and supply chain management, warehouse management has been paid more and more attention in logistics management center. Warehouse management has become the core of logistics management. At the same time, with the rise of e-commerce, the types and the turnover of goods have increased dramatically, resulting in the operational efficiency of warehouses playing a decisive role in the overall efficiency of the entire logistics system. With the in-depth application of IoT, many emerging technologies are changing the global industry pattern on an unprecedented scale, and, at the same time, they provide an opportunity for warehousing development to break through bottlenecks. The introduction and evolution of the concept of intelligent warehousing makes the development of warehousing technology have a broader research space. Therefore, the introduction of Internet of Things (IoT) technology into warehouse management is very meaningful.

Radio Frequency Identification (RFID) technology is a non-contact automatic identification technology which can automatically identify target objects and obtain relevant data through radio frequency signals. It has the advantages of high reliability, large storage capacity and easy modification. As the underlying key perception technology of IoT, it can quickly read and track electronic tags attached to items, and is widely used in positioning, monitoring, supply chain management and other applications. The maximum number of tags that can be detected simultaneously by a reader is different according to different RFID standards. For example, an OBID reader conforming to ISO 1693 standard can detect a maximum of 16 tags and some UHF RFID readers can detect 100 tags at the same time.

In the intelligent warehouse, the fixed RFID reader is used to identify the batch of materials and record relevant information. The inventory checking depends on the handheld RFID reader to check the inventory quantity. The warehousing logistics system uses the perception layer of IoT to identify materials, so as to obtain real-time warehousing production data. The information of intelligent warehousing is transmitted through the network layer to the application layer for processing. In addition, the application layer realizes the application interaction between warehousing materials and users, and between materials and materials.

RFID system is composed of readers and tags, when more than one tag replies to the reader at the same time, the collision occurs. The collision problem will seriously affect the integrity of tag data and make tags unable to be recognized correctly. The tag collision problem is one of the important problems in the RFID system, which greatly restricts the development of RFID technology. At present, the anti-collision algorithm has become one of the core topics in the theory and application of RFID. In this paper, an improved anti-collision algorithm (GMQT) is proposed and applied in intelligent warehouse to improve the identification rate of items and the turnover efficiency of warehouse.

2. Related Work

Currently, RFID anti-collision algorithms fall into two categories: tree-based deterministic algorithm and Aloha-based uncertain algorithm. The tree-based algorithm does not have the problem of label hunger, that is, there is no case that a label cannot be recognized for a long time, and the Aloha-based uncertainty algorithm has a label hunger problem due to random allocation of time slots. However, its advantage is that the algorithm is simple and easy to implement, and its performance is not affected by the length and distribution of label ID. According to the characteristics of warehousing, this paper will improve the tree-based anti-collision algorithm.

The query tree (QT) algorithm was first proposed by Law [1], and he set a stack in reader to store the query prefix. In every slot, the reader will broadcast the query command to the tags in the reader field, then the tags that match the query prefix 'q' respond to the reader, if multiple tags respond, there will be a collision, and the reader will push 'q0' and 'q1' into the stack, then the next query starts until the stack is empty. In the traditional tree traversal algorithm, regardless of whether or not there is a collision node of the tree, the algorithm will traverse the entire binary tree, which will lead to low efficiency in the RFID system. In order to solve the problems of long recognition time and low system efficiency in a tree-based deterministic algorithms, scholars have proposed many excellent improved algorithms in recent years.

Landaluce et al. [2–4] proposed a novel Query window Tree (QwT) protocol; the QwT effectively controls the number of responded bits from tags by adopting a dynamic-size 'window'.

Djeddou et al. [5] proposed a binary search algorithm based on continuous collision bits (IACA). If collision occurs, the reader can pick out the specific collision position according to the Manchester code, if the first and the second collision bits are consecutive, it will set the first two collision bits to '00', '01', '10', '11', respectively. Then, set the bits lower than the second collision bits to 1 and other bits remain unchanged. The reader set the new string as the command of next cycle. If the ID is less than command string, the tag will respond the reader with its own ID. However, if the first and the second collision bits are discontinuous, it will work as an ordinary binary search algorithm.

Jia et al. [6] proposed the collision tree (CT) algorithm that eliminates the idle cycle in tags identification absolutely.

Nikola et al. [7] studied the effect of tag ID distribution on tree-based memoryless algorithms and proposed a Flexible Query window Tree (FQwT) algorithm; it also reduced recognition time and energy loss through grouping labels by an estimated distributor.

Lai et al. [8] proposed an OQTT algorithm which divides the number of tags into several groups by optimizing segmentation criteria based on a bit estimation method and, finally, it uses a query tracking tree to identify tags quickly.

Zhang et al. [9] proposed an adaptive 4-ary pruning query tree algorithm (4APQT). Compared with the classical QT algorithm, it reduces the number of queries and unnecessary idle time slots, and improves the system throughput and communication complexity.

Fu et al. [10] designed a characteristic-value-based grouping rule and a collision bits rule to determine transmitted bit string combinations accurately and proposed a bit arbitration tree anti-collision protocol based on these two rules to decrease the time for collecting all tag IDs.

Shin et al. [11] proposed a novel tag anti-collision algorithm called a M-ary query tree scheme (MQT). A mapping rule is defined in the algorithm, and the m bits of the tag are mapped into 2^m bits and sent to the reader. In addition, the reader can deduce the collision bit combination according to the mapping rules. This algorithm can send data accurately and reduce idle time slots.

Although most of the above algorithms minimize collision and idle slots, the recognition efficiency is mostly between 0.5 and 0.55, and there is room for improvement. The efficiency of the MQT algorithm can reach 0.55 to 0.6 when m = 2. Although the bigger M is, the higher recognition efficiency is. However, due to the exponential growth of mapping strings, the amount of data transmission will also be greatly increased. Therefore, it is necessary to design an anti-collision strategy with higher recognition efficiency and less data transmission.

3. Group Mapping Query Tree Algorithm

3.1. Manchester Coding

Manchester coding is widely used for collision detection in tree-based algorithm [12–14]. In Manchester code, the value of a bit is defined by the change in level within a bit window. The negative transition means a bit '0', whereas the positive transition means a bit '1'. In an RFID system, if more than two tags transmit bits with different values, then the negative and positive transition will counteract. This result is impermissible in Manchester coding during data transmission. Therefore, Manchester code makes it possible to trace a collision to an individual bit and find where the collided bit is [15]. Supposing there are two tags, the principle of tracing collision bits with Manchester code is shown in Figure 1. The number of collision bits detected by the reader are 4 and 5. Manchester coding is also used in this paper to achieve the purpose of accurately detecting collision bits in the received data.

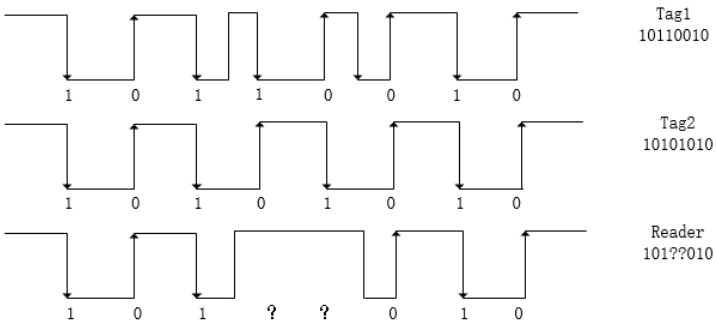

Figure 1. The sketch map of tracing the collision bits with Manchester coding.

3.2. Two Rules Definition

1. Group Rule Definition

Group mapping query tree algorithm (GMQT) uses a three-bit grouping strategy, that is, first, we process the three bits of the tag ID by XOR, and then group them according to the rules. In this way, not only the collision time slot can be reduced, but also the reader can accurately infer the combination of the transmission data from the collision information of the received data by using the collision bit inference method. The grouping idea in this paper is based on XOR operation of binary system. Each tag has a grouping counter GC to record the group number. For the tag i, assume that the three adjacent bits starting from the jth bit of tag i are $q_j q_{j+1} q_{j+2}$. In addition, the grouping counter GC(i) consists of two parts, the first one represents the value of $q_j \oplus q_{j+1}$, and the second one represents the value of $q_{j+1} \oplus q_{j+2}$. For example, for '$0_1 1_2 1_3$', there are $0_1 \oplus 1_2 = 1$ and $1_2 \oplus 1_3 = 0$, so GC = '10'. By analogy, it can be concluded that there are four possibilities for GC(i). That is, tags will be divided into four groups, as shown in Table 1. Since the combination of tags in each group may be known, the composition of the collision bits can be inferred from the group number.

Table 1. Grouping rule.

Group Number	Bits
00	000 111
01	001 110
10	100 011
11	010 101

2. Group Number Mapping Rule Definition

Since the group number still collides during the transmission process, the reader cannot clearly know which groups exist currently. For example, there are two groups '01' and '10', and the Manchester code of the group number obtained by the reader is '??'. Then, the reader will consider that all the group numbers exist, and sequentially query the '00', '01', '10', '11' groups, which will generate many idle slots, thereby reducing the recognition efficiency.

The group number mapping rule in this paper uses the mapping rule proposed by Shin [11], which helps the reader to determine the existing group number. The principle of mapping is to convert the k-bit data into a decimal number L, and then generate a 2^k-bits data, where the last L-bit is 1, and the remaining bits are 0. Because the length of group number is 2-bits, we set k = 2, and the mapping rules are shown in Table 2. The tag processes the group number according to the mapping rules, and then transmitted to the reader. Then, the reader can deduce the group number clearly according to the Manchester code of the mapping data. For example, there are two groups '01' and '10', After mapping, the two groups are '0010' and '0100', and the Manchester code obtained by the reader is '0??0'. Because of the collision of the 2nd and 3rd bits, we can deduce that the existing group numbers are '01' and '10' according to the mapping rules.

Table 2. Mapping rule.

Group Number	Mapping Code
00	0001
01	0010
10	0100
11	1000

3.3. Protocol Instructions

1. Query (PRE) represents the query instruction, PRE is the query prefix, and the tag whose ID matches the prefix PRE receives the instruction and responds to the reader query, and sends the new group number information and the remaining ID except PRE.
2. Query(PRE, G) represents the query instruction, PRE is the query prefix, and *G* is the group number, $G \in \{00, 01, 10, 11\}$. In addition, the tag whose ID matches the prefix PRE and group number is *G* is activated and participates in this query process.
3. PUSH(S, P) indicates read and write instructions, pushing P onto the bottom of the stack S.
4. P = POP(S) indicates a read and write instruction, and P pops up from the top of stack S.
5. ACK indicates a confirmation instruction, and the reader notifies the tag that it has been successfully identified and that the tag is placed in a dormant state.
6. '||' denotes a connector, which combines two sets of data into a group, such as '00 || 01' means '0001'.

3.4. Protocol Description

Based on the grouping strategy, we divide tags into four groups. The reader queries tags and receives ID and group number information from tags, infers possible groups according to Manchester code of the mapped group number, and then queries the possible group numbers in turn.

The flow chart of GMQT algorithm is shown in Figure 2. In addition, the procedures of this GMQT algorithm are as follows:

Step1: Initialization, the reader sets the query prefix stack PS and group number stack GS to be empty, and these stacks follow the first in first out principle.

Step2: The reader sends the command QUERY(prefix), and the query prefix is ε when the command is first sent, that is, all the tags within the reading range respond to the reader.

Step3: If there is no tag response, it means idle slot, and jump to Step 10; otherwise, jump to Step4.

Step4: If there is no collision, the reader sends a ACK command to identify the tag and puts it to sleep, then jump to Step10; otherwise, jump to Step5.

Step5: The reader receives decoded data 'pcode' of tags and mapping data 'gcode' of group, assuming that the structure of pcode is '$p_1p_2 \ldots p_{c-1}p_cp_{c+1} \ldots$', $p_i \in$0,1, and q_c is the first collision bit.

- If c > 3, set p_c to '0' and '1', respectively, and PUSH(PS, prefix || p1p2... Pc), then jump to Step10.
- If there is only one collision bit in $p_1p_2p_3$, set the collision bit to '0' and '1' respectively, and PUSH(PS, prefix||$p_1p_2p_3$); then, jump to Step10.
- Predict the possible group number $group_1 \ldots group_n$, and PUSH(GS,$group_1$)...PUSH(GS,$group_n$) in turn, and then jump to Step6.

Step6: group = POP(GS), and the reader sends command QUERY(prefix, group) .

Step7: If only one tag responds, the reader identifies the tag directly, and jump to Step9. Otherwise, jump to Step8.

Step8: According to the decode data the reader receives and the group, the reader can infer the bit string combinations as bc based on the rule of grouping, and PUSH(PS, prefix||bc).

Step9: When GS is empty, it means that all groups have been queried; then, jump to Step10. Otherwise, jump to Step6.

Step10: When PS is empty, it means that all tags have been identified. Otherwise, set prefix = POP(PS), and jump to Step2.

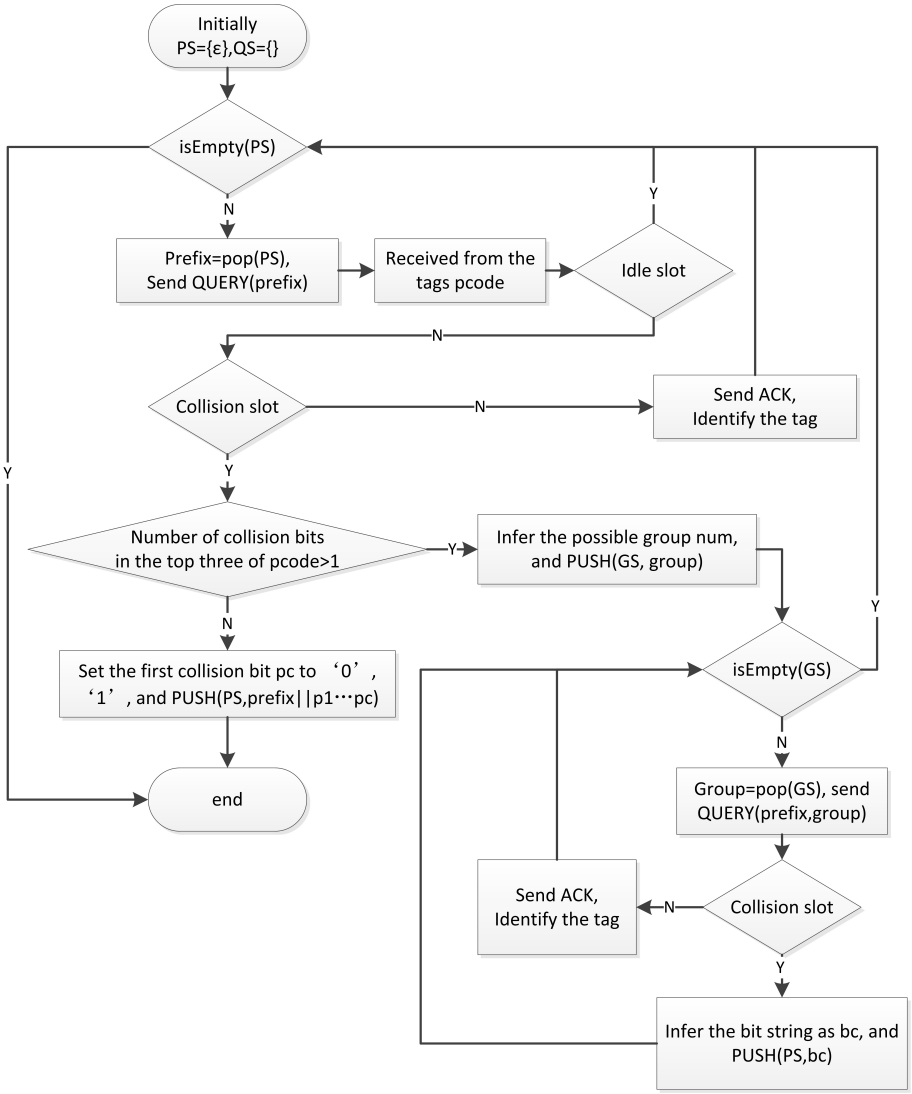

Figure 2. The flow chart of GMQT.

3.5. Example of GMQT

It is assumed that there are six tags: A(00010010), B(00001000), C(00110011), D(11011111), E(00100000), F(01001001). The details of the identification procedure with GMQT are shown in Table 3. Their identification procedure can be described as follows:

First slot: The reader broadcasts QUERY(ε) to all tags, and the tags transmit their IDs and group mapping data. The group mapping data the reader receives is ’?0??’; the reader can infer that there are three groups ’00’, ’01’, ’11’, and push them into GS.

Second slot: The reader pops new group ’00’ out from GS and broadcasts QUERY(ε,00); Tags A and B reply with their remaining IDs except prefix and the reader receives ’000??0?0’. There is no collision in the

top three bits. This means that there is only one possible combination '000' in group '00'. Thus, the reader sets the first collision bit to '0' and '1', respectively, and executes PUSH(PS,'0000') and PUSH(PS,'0001').

Third slot: The reader pops new group '01' out from GS and broadcasts QUERY(ε,01); Tags C, D and E reply and the reader receives '????????'. In the decode data, every bit is a collision bit, and it means that there are two possible combinations '001','110' of tags. The reader executes PUSH(PS,'001') and PUSH(PS,'110').

In the 4th, 5th, and 6th slots, Tags F, B and A respond, respectively, and then these tags are identified.

7th slot: The reader broadcasts QUERY(001). Tags C and E reply with their remaining IDs except '001' and the reader receives '?00??'. There is only one collision bit in the top three bits, so the reader sets the first collision bit to '0' and '1', respectively, and executes PUSH(PS,'001000') and PUSH(PS,'001100').

In the 8th, 9th, and 10th slots, Tags D, E and C respond, respectively, and then these tags are identified. The reader terminates the identification process when no prefix exists in the stack.

Table 3. The identification procedure of six tags.

Slot	QUERY		Tag Respond	Bit String			Stack	
	Prefix	Group		Status	id	Group Mapping	PS	GS
1	ε		Tag A,B,C,D,E,F	Collision	????????	?0??		00,01,11
2	ε	00	Tag A,B	Collision	000??0?0		0000,0001	01,11
3	ε	01	Tag C,D,E	Collision	????????		0000,0001,001,110	11
4	ε	11	Tag F	Identify F	01001001		0000,0001,001,110	
5	0000		Tag B	Identify B	1000	0100	0001,001,110	
6	0001		Tag A	Identify A	0010	0010	001,110	
7	001		Tag C,E	Collision	?00??	0?0?	110,001000,001100	
8	110		Tag D	Identify D	11111	0001	001000,001100	
9	001000		Tag E	Identify E	00		001100	
10	001100		Tag C	Identify C	11			

4. Performance Analysis

In an RFID system, the total number of slots for tag identification, system efficiency and communication complexity are important parameters in the anti-collision algorithm. Here, the performance of the algorithm is analyzed from these three aspects.

4.1. Total Number of Slots

Supposing that there are n tags in the system, and each tag has k bits. Because GMQT uses a three-bit grouping strategy, the search tree is an 8-ary tree. The root is in level 0 and the highest level of a tree is $\lceil k/3 \rceil - 1$ for a perfect 8-ary tree. Since the GMQT algorithm can accurately identify the group number and the collision bit, there is no idle slot, and the total number of slots $N_{cy} = N_{col} + N_{iden}$; here, N_{col} denotes the number of collision slot, and N_{iden} denotes the number of identification slots, so $N_{iden} = n$:

$$N_{col} = \sum_{H=0}^{\lceil k/3 \rceil - 1} \sum_{j=1}^{2^{3H}} P_{col}(H). \quad (1)$$

Because there is no idle slot, the probability of collision slots can be expressed as:

$$P_{col} = 1 - P_{iden}. \quad (2)$$

Let $P_{iden}(H)$ denote the probability that a node in level H is selected as an identification slot. $P_{iden}(H)$ corresponds to the probability that only one tag ID has the same prefix of $(k-3H)$ bits and $(n-i)$ tag IDs have different prefixes. Thus, $P_{iden}(H)$ can be derived as:

$$P_{iden}(H) = \binom{n}{1} \times \frac{\binom{2^{k-3H}}{1}}{\binom{2^k}{1}} \times \frac{\binom{2^k - 2^{k-3H}}{n-1}}{\binom{2^{k-1}}{n-1}}. \tag{3}$$

In addition to the slot of the query prefix, the group number needs to be queried separately, so the total number of slots can be expressed as:

$$G(n) = N(n) + N(group), \tag{4}$$

where $N(group)$ represents the number of slots for querying group number. After each command, the reader can infer the possible groups or prefixes directly according to the collision bits. Tags are divided into four groups at most, so we can get inequalities as follows:

$$0 \leq N(group) \leq 4N(n). \tag{5}$$

4.2. System Efficiency

The system efficiency, i.e., the output, is equal to the ratio of the number of successfully identified tags to the total number of queries taken to identify the tag. It is used to indicate how many queries the reader needs to successfully identify a tag. Reducing the total number of slots can effectively improve the system efficiency. Therefore, the system efficiency of GMQT algorithm is expressed as:

$$\eta = n/G(n). \tag{6}$$

4.3. Communication Complexity

Communication complexity refers to the number of bits that the reader needs to transmit to identify all tags in the system, and it also indicates the energy required by the reader. The more bits transmitted in the communication channel, the higher the energy consumption of the system.

Let $C(n)$ denote the communication complexity of GMQT, and $Q(n)$ denotes the total number of slots. $C(n)$ is expressed as:

$$C(n) = \sum_{i=1}^{Q(n)} (L_{reader,i} + L_{tag,i}). \tag{7}$$

Here, $L_{reader,i}$ denotes the number of bits the reader transmits for sending command in i_{th} cycles, and $L_{tag,i}$ denotes the number of bits the tags respond in i_{th} cycles. $L_{reader,i}$ consists of the length of reader transfer instruction $L_{com,i}$, the length of query prefix $L_{pre,i}$ and the length of query group $L_{group,i}$. $L_{tag,i}$ consists of the length of group mapping $L_{mapping,i}$, and the length of remaining IDs except prefix $L_{req,i}$. In GMQT algorithm, $L_{group,i} = 2$, $L_{mapping,i} = 4$. $L_{pre,i} + L_{req,i} = L_{ID}$, $L_{group,i}$ and $L_{mapping,i}$ don't exist every time. Only when the query command does not include group number do the tags reply with the group mapping code.

Thus, $P_{group} + P_{mapping} = 1$, and P_{group}, $P_{mapping}$ denote the possibility of $L_{group,i}$ and $L_{mapping,i}$.

In summary, C(n) is expressed as:

$$\begin{cases} C(n) = \sum_{i=1}^{Q(n)} (L_{com,i} + L_{ID} + 2P_{group} + 4P_{mapping}), \\ P_{group} + P_{mapping} = 1. \end{cases} \tag{8}$$

5. Simulation and Comparison

In this section, we compare GMQT algorithm to QT [1], 8-ary QT, IACA [5], BAT [10], MQT [11] from system efficiency and communication complex. We consider an ideal transmission channel, without capture-effect and path-loss affects. The RFID system, running on a MATLAB 2016a simulation platform, consists of a single reader and numerous tags, and the number of tags is increased from 5 to 1005. Each tag has a unique ID, and the length of tag ID is 16 bits.

1. System efficiency

Figure 3 shows the system efficiency with a different number of tags for GMQT, QT, 8-ary-QT, BAT, IACA, and MQT (m = 2). The system efficiency of GMQT is significantly better than other algorithms because GMQT reduces collision slots by grouping query, and it uses accurate prediction of group number collision bits by mapping rules to avoid idle time slots. Moreover, the system efficiency of GMQT doesn't decrease with the increase of the number of tags, and remained stable between 0.6 and 0.65. The second is MQT, whose output is between 0.55 and 0.6, and the output of BAT is about 0.5–0.55, and others are not more than 0.5. The mapping mechanism helps the GMQT algorithm avoid idle queries and the method of group query effectively reduce collisions, so when the number of tags increases, the efficiency of the algorithm will remain at a stable level.

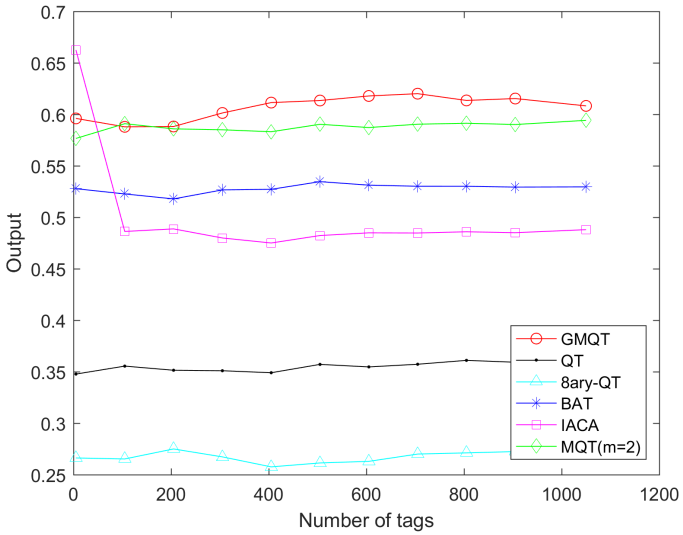

Figure 3. System efficiency with different protocols.

2. Communication complexity

Figure 4 shows the communication complexity with the different number of tags for GMQT, QT, 8-ary-QT, BAT, IACA, MQT (m = 2). The number of bits transmitted by GMQT is the least among the six algorithms. QT algorithm uses bit-by-bit identification to query tags, which consumes more slots, and for each slot the reader broadcasts the query prefix, the tag matching the prefix sends its ID to the reader completely. Therefore, using the QT algorithm, the number of bits transmitted in the communication channel is the largest and the communication complexity is the highest. The 8-ary-QT algorithm is similar to QT, and tags respond complete IDs to the reader. Compared with QT, 8-ary-QT reduces collision slots effectively, even though there are more idle slots, and there are only query instructions and prefixes transmitted by the reader in the idle slots, and no tags transmit data. Therefore, the communication

complexity of the 8-ary QT algorithm is better than that of the QT algorithm. Although MQT can accurately predict the collision bits to avoid idle slots, the mapping code returned by the tags which causes the transmission burden of the system. GMQT only maps the group number, which reduces the number of mappings. At the same time, the characteristics of the group also reduce the number of response tags. Therefore, GMQT can effectively reduce the communication complexity.

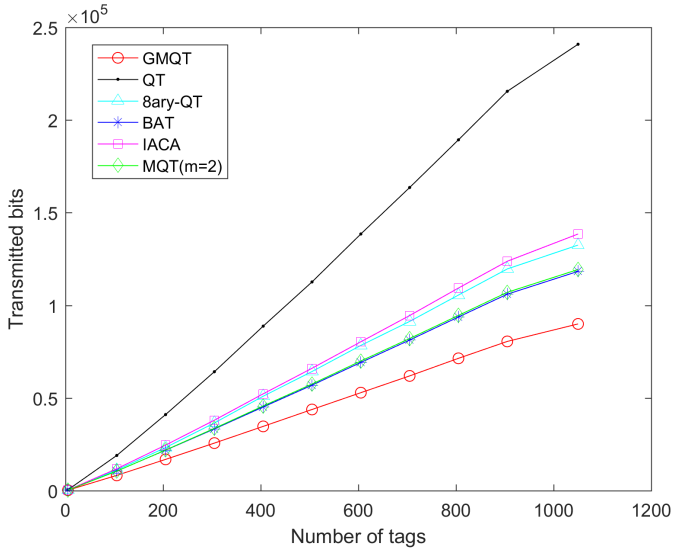

Figure 4. Communication complexity with different protocols.

6. Application

In the RFID system, the tag has the ID number of the unique global identification, which is attached to the surface of the identified item or embedded in the item [16]. The tags may store the name, price, date of production, manufacturer and other related information of the item. The reader identifies the tag on the item by transmitting the radio frequency signal and obtains the stored data information. Readers need to communicate with a large number of tags, and a single reader has only one communication channel. If there are multiple tags in the reader's reading range, each tag can only send information after the reader's command because the tag itself cannot perceive the existence of other tags. In the intelligent warehouse, there are a lot of tags waiting for identification during the inbound and outbound processes, and when two or more tags send their ID data to the reader at the same time, signal interference occurs, which results in the reader can not accurately distinguish information from these wireless signals, that is, the tag collision problem occurs, and the whole query process will be significantly delayed. Aiming at this problem, an improved anti-collision algorithm is proposed in this paper to reduce the occurrence of collision and idle time slots, thus improving the recognition efficiency.

In addition to attaching RFID tags to various goods, use RFID technology on the shelves, shelves, forklift trucks in the warehouse to make them have the ability of information feedback. The RFID reader is installed at the entrance and exit of the warehouse. When the goods enter and leave the warehouse in batches, the reader can get the information of the objects in and out of the warehouse automatically by reading the RFID tags attached to the goods. The forklift is equipped with a vehicle-mounted intelligent terminal and a hand-held RFID reader with wireless communication function, and the data is uploaded and sent through the wireless network and the back-end computer management system, and combined with the computer network technology, the warehouse operation state can be monitored in real time.

Throughout the process, all actions can be carried out in batches, which greatly improves the efficiency of the warehouse, because of the improved anti-collision algorithm. In addition, the layout of intelligent warehouse is shown in Figure 5.

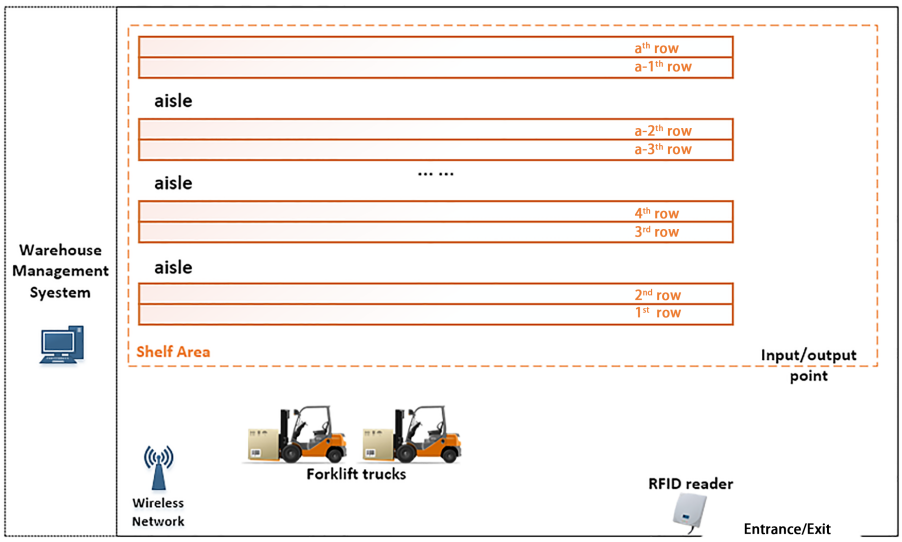

Figure 5. Layout of intelligence warehouse.

7. Conclusions

This paper improves the anti-collision algorithm based on grouping, and puts forward the GMQT algorithm. Comparing the GMQT algorithm with other algorithms such as QT, 8-ary QT, IACA, BAT and MQT in terms of system efficiency and communication complexity, the simulation results show that the GMQT algorithm has the best communication performance. The GMQT algorithm can effectively keep the recognition efficiency between 0.6 and 0.65, and reduce the communication complexity. Therefore, the GMQT algorithm can be applied to an intelligent warehouse, which can improve the operation efficiency of warehouse more effectively. However, most of the existing anti-collision algorithms are based on the fact that all the tags to be identified are initially within the reader recognition range, and no movement occurs until the recognition is completed. The case where a new tag moves into the reading range of the reader is not considered in the identification process, for example, in the warehouse with thousands of items, new items are put into the warehouse or other items are put into the wrong warehouse, and the item information is not recognized by the reader in time, that is, unknown tags appear in the system. If we use the traditional tag anti-collision algorithm to collect all tags, and then compare the tag information stored in the original database to identify unknown tags, the whole recognition process will consume a lot of time. Thus, next, we will study the dynamic tag recognition algorithm to improve the efficiency of identifying unknown tags.

Author Contributions: Conceptualization, Y.X.; Data curation, Y.X.; Formal analysis, Y.X.; Funding acquisition, L.Y.; Investigation, J.L. and D.L.; Methodology, Y.X.; Project administration, Y.X.; Supervision, Y.Q. and L.Y.; Validation, Y.Q., L.Y. and J.L.; Writing—original draft, Y.X.; Writing—review and editing, Y.X.

Funding: This research was funded by a "Research on Key Technologies of Intelligent Digital Mine Information Platform Based on Internet of Things" project of the Fujian Natural Science Foundation in 2019 Grant No. 2019J01011636 and "Intelligent Mine Construction and Industrialization Based on Internet of Things and Virtual Reality" project of the Fujian industrial area regional in 2019 Grant No. 2019H41010031.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Law, C.; Lee, K.; Siu, K.Y. Efficient memoryless protocol for tag identification: ACM. In Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Boston, MA, USA, 11 August 2000.
2. Landaluce, H.; Perallos, A. A fast RFID identification protocol with low tag complexity. *IEEE Commun. Lett.* **2013**, *17*, 1704–1706.
3. Landaluce, H.; Perallos, A.; Bengtsson, L. Simplified computation in memoryless anti-collision RFID identification protocols. *Electron. Lett.* **2014**, *50*, 1250–1252.
4. Landaluce, H.; Perallos, A.; Onieva, E. An Energy and Identification Time Decreasing Procedure for Memoryless RFID Tag Anticollision Protocols. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 4234–4247.
5. Djeddou, M.; Khelladi, R.; Benssalah, M. Improved RFID anti-collision algorithm. *AEU-Int. J. Electron. Commun.* **2013**, *67*, 256–262. [CrossRef]
6. Jia, X.; Feng, Q.; Yu, L. Stability Analysis of an Efficient Anti-Collision Protocol for RFID Tag Identification. *IEEE Trans. Commun.* **2012**, *60*, 2285–2294. [CrossRef]
7. Nikola, C.; Hugo, L.; Asier, P. Influence of the Distribution of Tag IDs on RFID Memoryless Anti-Collision Protocols. *Sensors* **2017**, *17*, 1891.
8. Lai, Y.C.; Hsiao, L.Y.; Chen, H.J. A novel query tree protocol with bit tracking in RFID tag identification. *IEEE Trans. Mob. Comput.* **2013**, *12*, 2063–2075. [CrossRef]
9. Zhang, W.; Guo, Y.; Tang, X. An efficient adaptive anticollision algorithm based on 4-ary pruning query tree. *Int. J. Distrib. Sens. Netw.* **2013**, *9*, 848746. [CrossRef]
10. Fu, Y.; Wang, X.; Wang, E. A bit arbitration tree anti-collision protocol in radio frequency identification systems. *Int. J. Distrib. Sens. Netw.* **2017**, *13*. [CrossRef]
11. Shin, J.; Jeon, B.; Yang, D. Multiple RFID Tags Identification with M-ary Query Tree Scheme. *IEEE Commun. Lett.* **2013**, *17*, 604–607. [CrossRef]
12. Liu, X.; Qian, Z.; Zhao, Y. An adaptive tag anti-collision protocol in RFID wireless systems. *China Commun.* **2014**, *11*, 117–127. [CrossRef]
13. Lai, Y.C.; Hsiao, L.Y.; Lin, B.S. Optimal slot assignment for binary tracking tree protocol in RFID tag identification. *IEEE/ACM Trans. Netw.* **2015**, *23*, 255–268. [CrossRef]
14. Chen, Y.; Feng, Q.; Zheng, M. Multiple-bits-slot reservation Aloha protocol for tag identification. *IEEE Trans. Consum. Electron.* **2013**, *59*, 93–100.
15. Chen, Y.H.; Horng, S.J.; Run, R.S. A Novel Anti-Collision Algorithm in RFID Systems for Identifying Passive Tags. *IEEE Trans. Ind. Inform.* **2010**, *6*, 105–121. [CrossRef]
16. Golsorkhtabaramiri, M.; Issazadehkojidi, N. A distance based RFID reader collision avoidance protocol for dense reader environments. *Wirel. Pers. Commun.* **2017**, *95*, 1781–1798. [CrossRef]