RFID for mobile applications

Arne Bosien, Volker Turau
Institute of Telematics
Hamburg University of Technology (TUHH)
Hamburg, Germany
{arne.bosien|turau}@tuhh.de

***Abstract*—The availability of fast anti-collision algorithms is crucial for most RFID applications. This paper aims to evaluate these algorithms for applications in which it is not intended to identify the entirety of moving objects but to detect as much tags as needed to allow orientation. The navigation of Automated Guided Vehicles (AGV) by distributed landmarks is an example which clarifies the discriminative requirements compared to supply chain tasks. For the former purpose redundant information can be gained from different tags. This requires the detection of an application dependent percentage of all tags. Because AGVs are moving, the detection and read and write operations have to be close together and very fast, since repetitive communication is not always possible.**

I. Introduction

In traditional RFID applications it has been required to detect all RFID tags in range. This may be useful if the determination of all products in a shopping cart is wanted, but this is not suitable for all imaginable uses.

The availability of fast anti-collision algorithms is crucial for most systems, but the employment in mobile applications makes higher demands on the communication speed because the reader is moving.

A. The meaning of anti-collision

To allow the communication with a specific RFID tag, all tags in the range of the RFID reader have to be identified at first. For this purpose the reader sends a single command which causes all tags to respond and send back their ID to the reader. Since the tags are not able to communicate with each other, the transmission of their responses leads to collisions, which inhibits the identification of the tags for the reader. To solve this challenge, anti-collision algorithms are required.

B. RFID for navigation purposes

Several approaches make use of stationary tags for orientation [Zam07], [Pec08] or navigation services of AGVs [BVT08], [NBOF06]. In this context it is not necessary to determine all tags by all means, and the speed of detection may overrule the importance of the completeness of an inventory if a high driving speed is necessary.

1) Example: Detection of one tag: To detect one single tag, which is exactly lying on the path of the reader, with a realistic detection time of $t_{\text{inv}} = 0.030\,s$ and a reader range of $r = 0.15\,m$ a maximum speed of

$$v = \frac{x}{t} = \frac{2r}{t_{\text{inv}}} = \frac{0.30\,m}{0.030\,s} = 10\,m/s$$

is possible. From the Nyquist-Shannon sampling theorem follows that at least two scans are necessary to detect the tag for sure. Therefore, the speed reduces to $v = 5\,m/s$.

2) Detection of more tags: Furthermore, the maximum speed is reduced if more tags are within the interrogation area. The number of requests to detect n tags with a simple anti-collision algorithm (described in section III-B) and incremented tag IDs can be calculated as the number of nodes of a binary tree. The height of this tree is given by $h = \log_2(n)$.

The time for a complete anti-collision cycle then can be estimated as:

$$\begin{aligned} t_{\text{inv}}(n) &= \left(2^{h+1} - 1\right) t_{\text{inv}} \\ &= \left(2^{\log_2(n)+1} - 1\right) t_{\text{inv}} \end{aligned}$$

The maximum speed for $n = 4$, $t_{\text{inv}} = 0.030\,s$ and $r = 0.15\,m$ is reduced to:

$$v = \frac{1}{2} \cdot \frac{2r}{\left(2^{\log_2(n)+1} - 1\right) t_{\text{inv}}} \approx 0.71\,m/s$$

3) Detection of boundary tags: As the reader is moving, the distance x, while a tag is in range of the reader, also depends on the offset y to the path (see Figure 1), which can easily be computed as:

$$x = 2\sqrt{r^2 - y^2}$$

Increasing the radius leads to a greater period when the detection is possible but unfortunately also to a higher amount of included tags if the tag density remains unchanged.

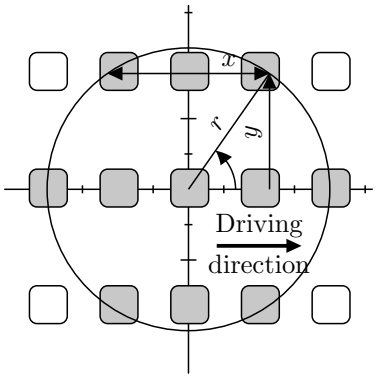

Fig. 1. RFID tags within the reader range

This underlines the special importance of fast anti-collision algorithms, especially if tags in boundary areas are also required to become detected.

C. Technologies

Several technologies exist to enable the usage of RFID in different environments. One criterion is the chosen operating frequency of the tags, which effects the distance wherein tags are accessible by the reader.

For the needs of navigating AGVs a range in the dimension of centimetres is wanted. These requirement are fullfilled by remote-coupling systems, that operate at distances between 1 and 100 centimetres and in the high frequency domain of 3-30 MHz. Furthermore writeable tags are available for these frequencies at a low price.

II. Stateless vs. Stateful

In the domain of high frequency RFID two fundamental different techniques are applied. They can be splitted into stateful and stateless anti-collision algorithms. Stateful algorithms require the tags to store information during the anti-collision process, while stateless algorithms have lower demands from the tags.

A. ALOHA (stateless)

ALOHA based anti-collision protocols are very simple protocols, which allow the tags to send their data whenever they want [Fin06]. The anti-collision is founded on the probability that two tags don't transmit at the same time. Slottet ALOHA enhances this procedure by introducing a number of defined slots, in which the tags may answer. Dynamic slotted ALOHA expands this by using a dynamic number of slots, which is adapted to the amount of involved tags.

B. Binary tree (stateful)

A binary tree anti-collision algorithm detects the tags by traversing the binary tree of possible tag IDs. For this purpose the reader transmits a sequence of bits.

After each received bit the tags compares if the corresponding bit of its own ID is matching. The tags maintain a pointer, which marks the current bit position in the ID. With every received bit the pointer is incremented. If the received bit mask is matching the ID, a tag sends the remaining bits of its ID to allow its identification. If more than one tag is responding, collisions are occurring, which prevents the identification of tags. If the bits are not matching the ID, a tag doesn't respond and quits its participation in the anti-collision cycle for now.

If collisions are detected, the reader is caused to send more bits until one tag can be identified or no tag is responding anymore. In both cases the reader sends another request command to reduce the bit mask to a shorter one. This causes quiet tags to participate again if the mask is short enough. This procedure is repeated until all tags are identified.

C. Query tree (stateless)

In query tree algorithms the reader sends out an inventory request with a part of a tag ID (mask, sometimes also called prefix), that causes all tags with matching ID to respond their whole ID. This may lead to collision. In the next step the reader extends the mask, which now matches to fewer responding tags. This procedure is repeated until only one tag is responding and hence is identified.

In each step the whole logic is performed by the reader. The tags just compare the received prefix with their ID and respond or not.

The advantage of a query tree algorithm is its simplicity on side of the tags and the ability to development better algorithms just by improving the algorithm performed by the reader without touching the behaviour of the tags.

III. Algorithms for query tree

A. Enabling technology

1) ISO-15693: The requirements for the following introduced algorithms are fulfilled by standards like ISO-15693for instance [ISO00]. The response to an inventory request is send by all tags at the same time. The transferred bits become superposed and even if collisions occur, the result may still be evaluated and used for the next improved inventory request.

This is achieved by using Manchester Coding.

2) Manchester Coding: Manchester Coding divides one bit into two signals. A high-to-low transition expresses a logical 0 and a low-to-high transition a logical 1. When two simultaneous transferred signals differ, the received data stream shows illegal parts which can be evaluated.

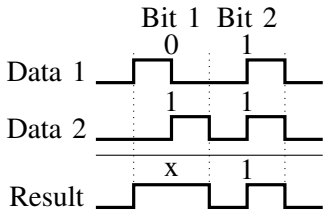

Fig. 2. Collision detection with Manchester Coding

3) Application to RFID: A collision can be caused by any number of tags, and it is not possible to determine the exact number of involved tags.

Tag 1 1001		
Tag 2 0101	Tag 1 1001	Tag 2 0101
Tag 3 1101	Tag 2 0101	Tag 4 1010
Result xx01	Result xx01	Result xxxx

The number of collision bits give information about the upper boundary of how many tags are involved. Since every collision is caused by at least two tags, n collisions are caused by maximum n^2 tags.

Some RFID readers just return the position of the most significant collision position, even if they detect the rest. In the following it is required that all collision positions are known.

B. Standard recursive algorithm

ISO-15693 presents a simple recursive algorithm for a complete inventory. This algorithms implements a depth-first search to identify all tags in range but offers many possibilities for improvement.

Tag	ID (Hex)	Binary
1	BA6	1010 1010 0110
2	A45	1010 0100 0101
3	F6B	1111 0110 1011
4	AAA	1011 1010 1010

As a first step an empty mask is sent, which makes all tags respond. In the second step the mask is extended to 0, which matches to tag 1 and tag 4. The renewed extension to 00 leads to no answer, since none of the tags ends with this bit sequence. Now this arm is left and in step 4 the mask consists of 10, which causes again tag 1 and tag 4 to respond. Transmitting of 010 leads to the identification of a single tag for the first time. Next 110 yields another tag. Now the 0-branch has been completed so that step 7 is going on with 1. The remaining steps are according to the previous ones and detect tag 2 and tag 3.

Step	Mask	Response	Meaning
1	ø	1x1x xxx0 xxxx	coll., extend mask
2	0	101x 1010 xx10	coll., extend mask
3	00	ø	no response
4	10	101x 1010 xx10	coll., extend mask
5	010	1011 1010 1010	**Tag 4 detected**
6	110	1010 1010 0110	**Tag 1 detected**
7	1	1x1x 01x0 xxx1	coll., extend mask
8	01	1010 0100 0101	**Tag 2 detected**
9	11	1111 0110 1011	**Tag 3 detected**

C. Tag-ID-design with parity bit

The following approach is a combination of anti-collision algorithm and the selection of tag IDs [KKLA08]. It makes the assumption that the least significant bit of the tag ID is a parity bit. When this is given, in case of two remaining collision two tags can be identified at the same time.

1) Identify two tags at the same time at one collision: If a single collision occurs, it is possible to identify two tags at once. Just the IDs of tag 1 and tag 2 exist to explain the result of the collision.

Tag 1 1001
Tag 2 1101
Result 1x01

2) Identify two tags at the same time at two collisions: In the next example the least bit is a parity bit whereas a 1 indicates an even number of ones in the tag ID.

Tag 3 0001
Tag 4 1011
Result x0x1

As in section before this leads in a first step to two possible tag IDs: x001 and x011

But because the last bits indicates the number of ones, it can be concluded that the first tag ID is 0001 and the second one 1011.

While this speeds up the inventory it reduces the number of possible IDs by the factor of 2. For application like suggested in [BVT08] it may be sufficient if the ID is unique during a certain area and not necessarily across the whole operating area. This would allow the reuse of tag IDs in case availability of unused IDs is getting short.

D. Setting tags quiet

If a tag is not only to be inventoried but also to be written or read, an interesting possibility of nesting inventory and read/write operations appears [Fin06]. As soon as a tag has been found during the normal anti-collision cycle, the tag will be selected with a first command. The next command is for read or write operations, and the last one puts the tag into the sleep state. Thus the tag will not take part in the anti-collisions anymore, even if a matching mask has been sent.

The anti-collision algorithm could use this approach to continue with shorter mask, since now it is not more required to exclude the previous found tag from responding to inventory request. This technique is only reasonable if tags are to be written or read indeed. In any other case putting a tag to sleep takes two much time due to the fact that the whole tag ID has be transferred. Only by combining select, read/write and sleep commands this approach becomes meaningful.

Despite the slight improvement this approach may offer to the inventory speed of one anti-collision cycle, the main advantage is shown when more inventories are executed in succession. Because in the next cycle previously inventoried tags don't appear again, and only new tags have to be identified. Of course this is an disadvantage if the information is wanted if a tag is still in the reader's range or not.

E. Remember masks

Myung and Lee propose another approach to improve the speed of repeated inventories [ML05]. The algorithms stores the masks, which cause only one tag to respond (identified node) and those, which cause no tag to respond (no-response node). In the following cycle the stored masks are proved. New tags either collide with a recurring tag or make no-respond nodes become identifying nodes. The former case requires a new anti-collision process for this part of the tree. If instead a tag leaves the field, an unidentified node becomes a no-response node and can possibly be merged with another no-response node.

Similar to the approach of the previous section this approach reduces the time of repeated inventories when the involved tags don't change or just to a small amount. But in contrast all tags currently in the reader's range are detected. A drawback can be expected when many tags are changed, because then storing of old masks doesn't pay off and leads to many useless requests.

IV. Improvements

A. General improvements

This section describes general improvements, which can be combined with the algorithms introduced in section III.

1) Omit empty branches: The first very obvious improvement is to omit apparent unnecessary request.

In section III-B transmitting `00` in step 3 leads to no responding tags. A closer look on the response of step 2 shows that there are no collisions in the last two bits (`xx10`) and that the second bit is a `1`. Therefore, step 3 can be omitted, which test for known not existing tags.

2) Jump to collision position: The second improvement is similar to the previous one and omits request that leads to same results as previously performed requests.

This can be examined in step 4 which leads to the same response as step 2. Since step 2 is known that the last two bits don't collide, so it is not necessary to prove these bits again. Step 4 also can be omitted.

3) Don't start with empty query: Can be assumed that many tags are within the readers range, it is reasonable not to start with an empty query since this will lead to collisions with a very high probability. If an uniform distribution of tag IDs can be supposed, the probability that the IDs of three tags don't lead to a collision in the last bit is $2 \cdot 0.5^3 = 0.25$, which means that in 75% of all cases a collision can be expected. This makes it meaningful to start with the queries `0` and `1`. The more tags can be expected the start query can be extended to `00`, `01`, `10`, `11` etc.

Of course this approach may lead to failed request if no tag fits the supposed initial query.

But it should be noticed that the execution of a request that leads to no responding tags is faster than a request with responding tags, because no transferred IDs have to be received.

4) Extension of mask: Furthermore it can be considered to apply the approach of the section before not only to the start query but also to the ongoing anti-collision. [HYHH08] presents a simple approach that adapts the extension of the mask during the anti-collision cycle to the remaining bits of the tag ID. The length of the mask is calculated as $R_{PB} = \sum_S^K \frac{N}{2^S}$. N is the overall length of the tag ID, $S \leq N$ is the level of the query tree and $K = \log(N)$.

Another approach could be to adapt the extension not to the remaining bits but to the occurred collisions and the herewith estimated remaining tags.

B. Improvements for moving objects

Previously presented algorithms are suitable for the shopping cart scenario. For navigation of AGVs it is not necessarily required to detect all tags. Since an AGV is moving, the detection speed has a greater importance, and it is also imaginable that the detection of only very few tags is sufficient.

1) Guess arms and compare collisions: If the results of step 1 and step 2 (section III-B) are compared, it can be concluded that there is at least one tag that ends with `01` since the `0`-arm leads to collisions in bit 3 and bit 4 but not in bit 2. Therefore, step 7 could be committed and step 8 executed directly.

Now comparison between steps 1, 2 and 8 shows that not all collision of the response of step 1 can be explained (watch 11th bit). This means that at least one other tag exists in the `11`-arm.

Unfortunately even if all collision could be explainable, it can not be eliminated that there are remaining undiscovered tags. This approach can be considered if speed is more important that the completeness of the inventory.

2) Remember n-bit masks: The algorithm presented in section III-E is useful for repetitive inventory processes in a static environment. If tags leave the interrogation area, this will cause request which don't generate responses [Kha09]. This issue can not be avoided generally, but it can be moderated by limiting the length of the stored mask depending on the number of found tags.

3) Select tag IDs: It can be observed that the inventory speed not only depends on the number of tags but also on the involved tag IDs. While a collection of the tags

Tag 1: `0111` Tag 2: `0101`
Tag 3: `0110` Tag 4: `0100`

can be detected very fast (Figure 3) and requires short masks lengths of at most 2 bits, the inventory of the following tags requires more anti-collision steps:

Tag 1 `1111` Tag 2 `1011`
Tag 3 `1001` Tag 4 `1000`

On the other hand one tag (`1000`) can be detected very fast and by transmission of just a one bit mask (`0`).

For AGV navigation the inventory is always performed for stationary and proximate tags. This fact can be exploited to enhance the inventory speed by putting only tags into neighbourhood whose IDs are easy to detect.

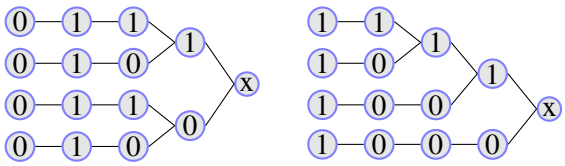

Fig. 3. Balanced and unbalanced tag ID tree

Furthermore the IDs can be selected in a way that they differ preferably in few bits. This accelerates the inventory process regarding to section III-C1.

C. Improvements for special tag distributions

1) Estimate tag quantity in arms: In the following it is supposed that a method exists to estimate a more or less correct number of tags in an arm. This maybe can be achieved by analysing the number of collisions and a good knowledge of the tag ID distribution.

Then an algorithm could be designed to take advantage of strong unbalanced structures like in Figure 3 and handle short arms at the beginning to find first tags very fast. This does not necessarily improve the duration of a complete anti-collision cycle but it minimises the average tag detection time.

Such an algorithm could perfectly be used if not all tags are needed to be detected and can abort if a defined number of tags is found.

2) Exploitation of balanced ID trees: An approach to exploit an ID distribution of sequential IDs, which leads to a more or less balanced tree (Figure 3), is given in [KYK08] and called *Sidewalk*. This make use of the observation that allmost all identifying nodes are at the same level.

Once the depth of the tree has been determined by breadth-first search, this algorithm checks for tags at all other arms at the same depth. Generally this may lead to requests without responses but this is accepted. The presented algorithm also works for other ID distributions but reaches its best performance for sequential IDs.

V. Simulation results

The most important algorithms have been simulated with different tag ID distributions. None of the simulated algorithms is designed to make use of the knowledge how the IDs are distributed.

A. Speed simulation

For the simulation of the behaviour of algorithms executed on a moving RFID reader, the three following distributions have been used:

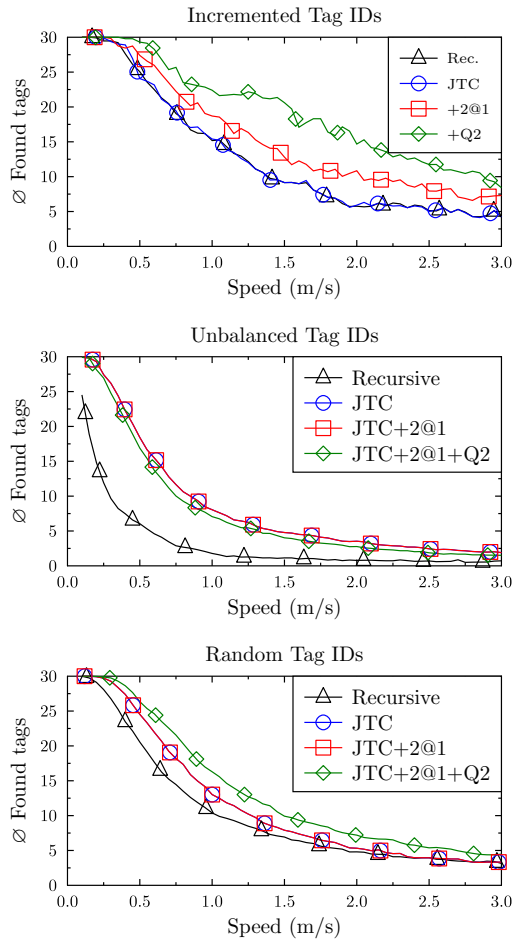

Fig. 4. Recursive algorithms with several Tag ID distributions

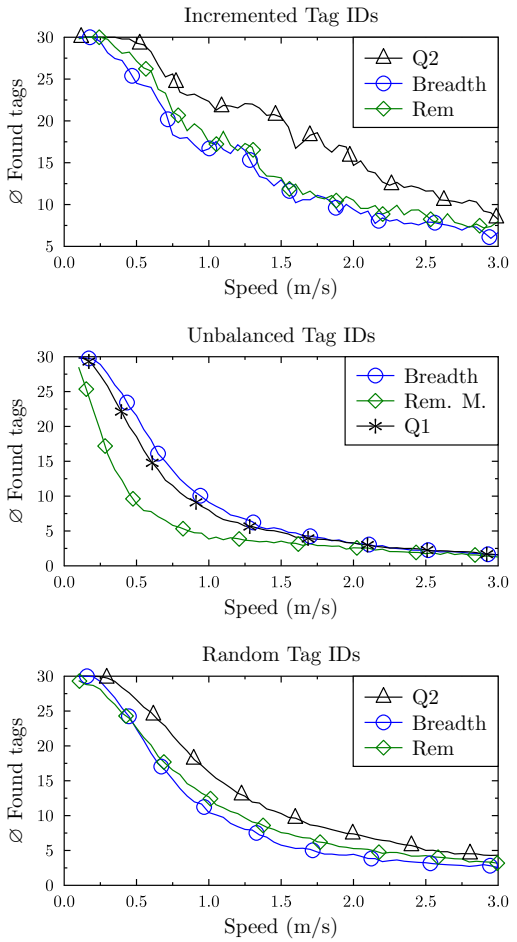

Fig. 5. Various algorithms with several Tag ID distributions

Incremented Tag IDs The tag IDs are counted up. This yields to a balanced and dense tree. The initial ID is generated randomly.

Unbalanced Tag IDs The IDs of the tags are selected in such a way, that a strong unbalanced tree is build. The initial ID is generated randomly. All other IDs contain the same lower bit pattern of random length.

Random Tag IDs The whole ID of the tags are generated randomly, which results in a balanced tree.

30 tags have been placed in a grid of 3x10 tags with a length of $1\,m$ and a width of $0.25\,m$. The RFID reader moves with a constant velocity over the grid and the inventory is continuously repeated. The range of the reader has been set to $0.14\,m$. This means that up to six tags are at the same time inside the range of the reader. Each simulation has been run 100 times and the result has been averaged.

Figure 4 shows the result for different modification of the standard ISO-15693 algorithm (see section III-B). Because of the nature of the simulation (the time dependent positions of the reader is always the same) the resulting curve is very rough when incremented tags ID are simulated. Here jumping to collision positions (JCP) doesn't enhance the performance. Identifying two tags at the same time (2@1) gives a notable advantage of up to 88% (at a speed of $2\,m/s$). Using an additional initial query (Q2) with four masks (`00`, `01`, `10`, `11`) leads to another sizable improvement of up to 188% ($v = 1.95\,m/s$) compared to Rec/JTC.

With unbalanced tags, the main improvements can be observed by jumping to the collision position. Initial query masks lower the performance.

The best performance with random tag IDs is again reached by using initial query masks. As before, the 2@1 modification doesn't pay off.

Figure 5 show result of the simulation of different anti-collision approaches, including Breadth First Search and an algorithm that remembers previously detected tags. It can be observed that for incremented and random tag IDs again Q2 gives the best performance.

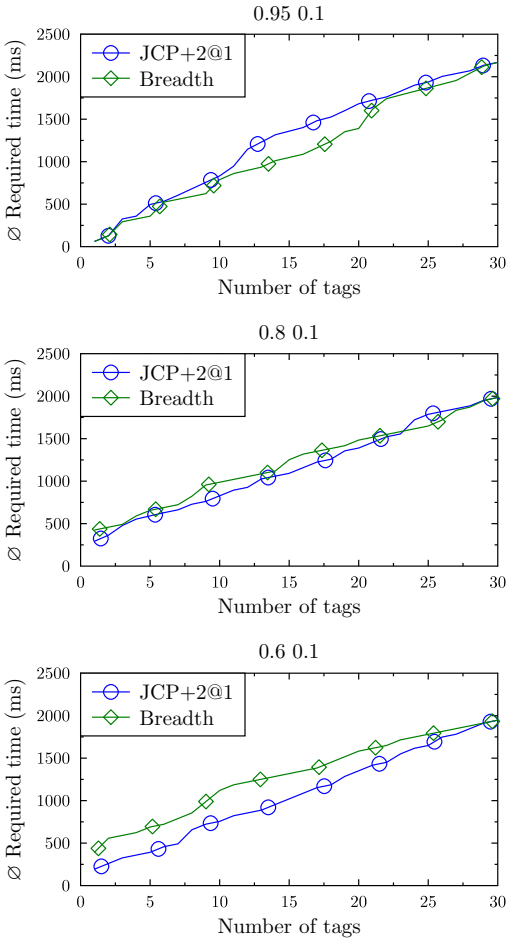

Fig. 6. Breadth First Search and Depth First Search algorithms with different tag ID distributions

B. *Unbalanced distributions*

Figure 6 shows the different behaviour of Breadth First and Depth First Search algorithms and time which is needed to detect 30 Tags with unbalanced tag ID distributions. Each simulation has been run 1000 times.

Both algorithms have the same execution time, but it can be seen that down have a constant detection rate. The Breadth First Search has its benefits for higher p. Balanced distributions (lower p) are the domain of Depth First Search algorithms.

VI. Conclusion

The possible driving speed of AGVs is directly linked to the speed of the inventory and the tag communication. Therefore, fast anti-collision algorithms are required.

The inventory algorithms can take advantage of all optimisations suggested to the inventory of classical RFID application, but moreover new optimisation possibilities are arising. It was shown that the inventory speed is connected to the distribution of involved tag IDs. While normally just a limited influence on this distribution is given, in the suggested application the tags are stationary and fixed, which offers a greater control over tags in the reader's range. Therefore, the selection of ideal tag IDs appears as a new challenge.

Furthermore, it is not longer required to detect all tags. Here, to shorten the time until first tags can be identified gains more importance.

In respect to this, new criteria for measuring the efficiency of algorithms are needed, because common anti-collision algorithms try to optimise the overall inventory speed.

Not discussed here is the employment of slotted query tree algorithms, which offers capabilities especially for balanced ID structures.

References

[BVT08] Arne Bosien, Marcus Venzke, and Volker Turau. A rewritable RFID environment for AGV navigation. In *Proceedings of the 5th International Workshop on Intelligent Transportation (WIT'08)*, Hamburg, Germany, March 2008.

[Fin06] Klaus Finkenzeller. *RFID Handbuch*. Carl Hanser, 4. edition, 2006.

[HYHH08] Ching-Hsien Hsu, Chia-Hao Yu, Yi-Pin Huang, and Kyung-Jae Ha. An enhanced query tree (eqt) protocol for memoryless tag anti-collision in rfid systems. volume 1, pages 427–432, Dec. 2008.

[ISO00] ISO/IEC. Iso/iec fcd 15693-3 part 3: Anti-collision and transmission protocol, 03 2000.

[Kha09] Sascha Khan. Optimization of rfid anti-collision algorithms. Master's thesis, Hamburg University of Technology, May 2009.

[KKLA08] SungSoo Kim, YongHwan Kim, SeongJonn Lee, and KwangSeon Ahn. An improved anti collision algorithm using parity bit in rfid system. volume Seventh IEEE International Symposium on Network Computing and Applications. IEEE, 2008.

[KYK08] Hyunho Koh, Sangki Yun, and Hyogon Kim. Sidewalk: A rfid tag anti-collision algorithm exploiting sequential arrangements of tags. In *ICC*, pages 2597–2601, 2008.

[LXXL] Leian Liu, Zhenhua Xie, Jingtian Xi, and Shengli Lai. An improved anti-collision algorithm in RFID system. Technical report, South China University of Technology.

[ML05] Jihoon Myung and Wonjun Lee. An adaptive memoryless tag anti-collision protocol for rfid networks. 2005.

[NBOF06] Björn Niemann, Mathias Baum, Ludger Overmeyer, and Dirk-H. Fricke. Aufbau von fahrerlosen transportsystemen (fts) durch eine dezentrale datenstruktur. *Logistics Journal*, 2006.

[Pec08] Morgen E. Peck. RFID Tags Guide the Blind. *IEEE Spectrum*, 1, 2008.

[Zam07] Marco Mameiand Franco Zambonelli. Pervasive pheromone-based interaction with rfid tags. *ACM Trans. Auton. Adapt. Syst.*, 2(2):4, 2007.