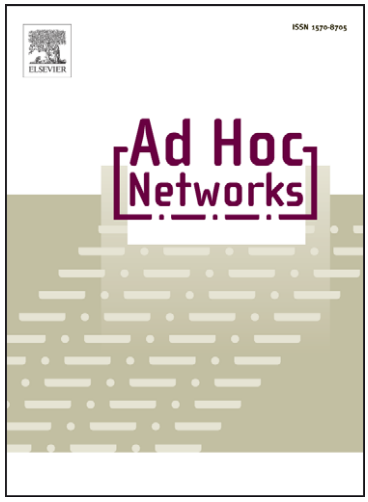

Instant Collision Resolution for Tag Identification in RFID Networks

Maurizio A. Bonuccelli, Francesca Lonetti, Francesca Martelli

PII: S1570-8705(07)00040-6
DOI: 10.1016/j.adhoc.2007.02.016
Reference: ADHOC 242

To appear in: *Ad Hoc Networks*

Please cite this article as: M.A. Bonuccelli, F. Lonetti, F. Martelli, Instant Collision Resolution for Tag Identification in RFID Networks, *Ad Hoc Networks* (2007), doi: 10.1016/j.adhoc.2007.02.016

Instant Collision Resolution for Tag Identification in RFID Networks

Maurizio A. Bonuccelli [a,b] Francesca Lonetti [a,b] Francesca Martelli [a]

[a] *Dipartimento di Informatica, Università di Pisa*
Largo Pontecorvo 1, Pisa, Italy
[b] *Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" (CNR)*
Via Moruzzi, Pisa, Italy

Abstract

In this paper, we approach the problem of identifying a set of objects in an RFID network. We propose a modified version of Slotted Aloha protocol to reduce the number of transmission collisions. All tags select a slot to transmit their ID by generating a random number. If there is a collision in a slot, the reader broadcasts the next identification request only to tags which collided in that slot. Besides, we present an extensive comparative evaluation of collision resolution protocols for tag identification problem in RFID networks. After a quick survey of the best performing RFID tag identification protocols, both deterministic and probabilistic, we present the outcome of intensive simulation experiments set up to evaluate several metrics, such as the total delay of identification process and the bit complexity of reader and tags. The last metric is strictly related to energy constraints required by an RFID system. The experiments point out that our protocol outperform all the other protocols in most cases, and matches them in the others.

Key words: Radio Frequency Identification Networks, Collision Resolution, Medium Access Control, Aloha Based Protocols, Tree Based Protocols

1. Introduction

Fast and reliable identification of multiple objects that are present at the same time is very important in many applications. A very promising technology for this purpose is Radio Frequency Identification (RFID), which is fast pervading many application fields, like public transportation and ticketing, access control, production control, animal identification, localization of objects and people. An RFID system consists of radio frequency (RF) tags attached to objects that need to be identified and one or more networked electromagnetic readers. The great appeal of RFID technology is that it allows information to be stored and read without requiring either contact or a line of sight between the tag and the reader. For this contact-less feature, RFID technology is an attractive alternative to bar code in the distribution industry and supply chain, since it can hold more data.

In RFID systems, tags can be active or passive. Active tags have storage capabilities and are provided with power sources for computing and transmitting data. Due to the complexity and cost of mounting a power source onto a tag, active tags are not practical for use with disposable consumer products. Passive tags instead rely only on RF energy induced by the electromagnetic waves emitted by the reader, and can have limited storage functionality. In a typical communication sequence, the reader emits a continuous radio frequency wave. When a tag enters in the RF field of the reader, it receives energy from the field, for modulating the signal according to its stored data. Due to more advanced protocols

Email addresses: bonucce@di.unipi.it (Maurizio A. Bonuccelli), lonetti@di.unipi.it (Francesca Lonetti), f.martel@di.unipi.it (Francesca Martelli).

and circuit design development, the reliability and the read range of passive RFID networks continue to improve, and their cost continue to decrease, which leads to an increase of passive tags applications.

A very important issue in RFID systems with passive tags, is complexity, and computing and transmitting capacity of tags in the identification process. The reader, containing internal storage capacity and processing power, broadcasts a request message to tags asking for the unique tag ID, or detailed information saved in them. After receiving this message, all or some tags send their response back to the reader. If only one tag answers, the reader receives just one message which is correctly decoded. If two or more tags answer, their messages will collide on the RF communication channel and cannot be correctly received by the reader. An effective system must avoid, or at least limit, these collisions by using anti-collision protocols in the identification process. In this paper, we present a new protocol for limiting such collisions, and evaluate the complexity of reader and tags required by our and several different tag identification protocols. The evaluation is about communication complexity in terms of number of reader queries and bits sent by reader and tags. Another commonly used metric that we evaluate is the number of time units needed to identify all tags, that is strictly related to delay of the identification process. This evaluation is very useful when designing an RFID system to foresee energy constraints and resource requirements.

Thus, a very important issue in RFID applications is the fast and reliable identification of multiple tagged objects simultaneously, assuming that the exact number of tags is not known in advance. This is a special case of multiple access communication problem that has been studied extensively in the past. The solution to this problem is given by different collision resolution protocols [1]. However, in RFID applications, the problem is more challenging and complex, because of the memory and energy constraints of tags [2–7]. We do not assume that passive tags can have sensing capabilities to detect collision, or that they can communicate with other tags directly, since such assumtions are not realistic. In [3] tags are assumed to have collisions detection capability, which implies a more expensive and power consuming hardware. In that paper, it is also assumed that the reader knows the exact number of tags to be identified. These assumptions help in solving the collision problem in tag identification process but are not realistic. The purpose of this paper is twofold: We present a new tags identification protocol, and we compare the performance of the best performing protocols presented so far. This last aspect of the paper is very useful since no such comparison has ever been made: When a new protocol has been proposed, it has been compared with at most one other protocol. As we shall see in the paper, the results of our comparison substantially confirm the performances that were originally presented along with the protocols.

Table 1
Pseudo-code of BS protocol

```
Reader procedure:
1.  channelStatus=2;
2.  while(channelStatus> 1) do
3.      broadcast(query);
4.      receiveAnswers;
5.      broadcast(channelStatus);
6.      if (channelStatus== 1) then tagIdentification();

Tag procedure:
11.     identified = false; myCounter = 0;
12.     while (not identified) do
13.         receive(query);
14.         if (myCounter == 0) then sendAnswer;
15.         receive(channelStatus);
16.         if (channelStatus > 1) then
                if (myCounter == 0) then
17.                 myCounter+ =rand()%2;
18.             else myCounter + +;
19.         if (channelStatus <= 1) then
20.             myCounter − −;
21.             if receivedIDrequest then
22.                 send myID;
23.                 identified = true;
```

The paper is organized as follows. In Section 2, we survey the main known protocols for the tag identification problem, with more details for the behavior of those selected to be evaluated (by means of simulation trials) in this paper. In Section 3, we describe our approach, by highlighting the assumptions we made. In Section 4 we present simulation setting, while in Section 5 we show the results of our simulative evaluation. In Section 6, conclusion and future work complete the paper.

2. Related Work

As usual in medium access control problems, the proposed protocols for collision resolution in RFID systems are either *probabilistic* and *deterministic*. The first ones are *Aloha-based* protocols, the last ones are *Tree-based* protocols. There are also hybrid approaches, where randomization is applied in tree

Table 2
Pseudo-code of DFSA protocol

Reader procedure:
1. $l_0 = N_{EXP_0}$; $c_k = l_0$;
2. **while** ($c_k > 0$) **do**
3. $c_0 = 0$; $c_1 = 0$; $c_k = 0$;
4. broadcast(l_i);
5. **for** $s = 1$ **to** l_i **do**
6. receiveAnswers;
7. **if** (channelStatus[s]= 1) **then**
8. *tagIdentification();*
9. c_1++;
10. **if** (channelStatus[s]> 1) **then** c_k++;
11. **if** (channelStatus[s]= 0) **then** c_0++;
12. N_{EXP_i} =*ChebyshevEstimation*(l_i,c_0,c_1,c_k);
13. l_{i+1}=$N_{EXP_i} - c_1$;

Tag procedure:
14. $identified = false$;
15. **while** (**not** $identified$) **do**
16. receive(l_i);
17. s = randomNumber mod l_i;
18. sendAnswer in slot s;
19. receiveMsg;
20. **if** $Msg = IDrequest$ **then** send $MyID$;
21. $identified = true$;

Table 3
Pseudo-code of AFSA protocol

Reader procedure:
1. $l_0 = 128$; $group = 1$; $c_k = l_0$;
2. **while** ($c_k > 0$) **do**
3. $c_0 = 0$; $c_1 = 0$; $c_k = 0$;
4. broadcast($l_i, group$);
5. **for** $s = 1$ **to** l_i **do**
6. receiveAnswers;
7. **if** (channelStatus[s]= 1) **then**
8. *tagIdentification();*
9. c_1++;
10. **if** (channelStatus[s]> 1) **then** c_k++;
11. **if** (channelStatus[s]= 0) **then** c_0++;
12. N_i =*ChebyshevEstimation*(l_i,c_0,c_1,c_k);
13. $N_{i+1} = N_i - c_1$;
14. set l_{i+1} and $group$ according to Table 4;

Tag procedure:
15. $identified = false$;
16. **while** (**not** $identified$) **do**
17. receive($l_i, group$);
18. s = randomNumber mod l_i;
19. **if**($group > 1$) **then**
20. g = randomNumber mod $group$;
21. **if** ($g = 0$) send $myID$ in slot s;
22. **else** sendAnswer in slot s;
23. receiveMsg;
24. **if** $Msg = IDrequest$ **then** send $MyID$;
25. $identified = true$;

schemes [3,8].

In Slotted Aloha based identification protocols, the time is assumed to be slotted and all tags have a local clock for synchronization. A time slot is a time interval in which tags transmit their serial number or a detailed information saved in them. A read cycle is a tag identifying process that consists of a frame. A frame is a time interval between requests of a reader and consists of a number of slots. Each tag transmits its serial number to the reader in a slot of a frame, and the reader identifies the tag when a time slot is used by one tag only. To simplify, it can be assumed that one slot consists of two subslots, one for probes/response from reader and one for transmissions from tags. The basic Framed Slotted Aloha protocol [9] uses a fixed frame size and does not change the size during the process of tag identification. In this protocol, the reader transmits to tags the frame size and each tag generates a random number j not larger than the frame size, and then transmits in the j^{th} slot of the frame. With a fixed size of the frame, if there are too many tags, most slots will have a collision. On the contrary, there are many wasted time slots, if a large size of the frame is used with a small number of tags. The Dynamic Framed Slotted Aloha (DFSA) [6,10] protocol changes the frame size dynamically. We evaluate the performance of a DFSA protocol that sets the proper frame size of each read cycle by using Chebyshev's inequality, which says that the outcome of a random experiment involving a random variable X, is very likely close to the expected value of X. Let c_0, c_1, c_k represent the number of empty slots, of slots filled with one tag transmission, and of slots with collision, obtained in a frame, respectively, and let a_0, a_1, a_k be their expected values. So, by minimizing the difference between them, we obtain an estimation of the number of tags which have transmitted in the last frame. Since our protocol too uses Chebyshev's inequality, we shall describe it in Section 3.3. Pseudo-code of DFSA protocol is reported in Table 2. N_{EXP_i} represents the expected number of unread tags at the beginning of frame i of length l_i; the value of channelStatus[s] can be zero, one, or more than one, representing no tag transmission in slot s, one transmission, or colliding transmissions, respectively.

The constraint of this protocol is that the frame size cannot be increased indefinitely as the number of tags increases, but it has an upper bound. This implies a very high number of collisions when the number of tags exceeds the maximum admitted

Table 4
Frame size adjustment for AFSA and LTSA

Estimated unread tags (N_i)	Frame size (l_i)	Groups
1 - 11	8	1
12 - 19	16	1
20 - 40	32	1
41 - 81	64	1
82 - 176	128	1
177 - 354	256	1
355 - 707	256	2
708 - 1416	256	4
...	...	...

frame size. Such problem is approached in [7], where an enhanced dynamic Framed Slotted Aloha protocol is proposed. We evaluate the performance of this protocol too, that we call Advanced Framed Slotted Aloha (AFSA). It works as follows: The size of the first frame is always set to 128. The size of the next frames is set according to Table 4. When the number of unread tags is estimated to be greater than 354, the reader divides the tags in groups, and chooses the number of groups that maximize the system efficiency function given by the ratio between the number of slots filled with one tag transmission and the current frame size [7]. The reader then broadcasts the number of groups and tags use this number for a modulo operation on a randomly generated number. Only tags for which the modulo is zero will transmit in that frame. Pseudo-code of AFSA protocol is reported in Table 3.

Tree-based tag anti-collision protocols can have a longer identification delay than Slotted Aloha based ones, but they are able to avoid the so called tag starvation, in which a tag may not be identified for a long time when involved in repeated collisions. Among tree-based protocols, there are binary search protocols [11] and query tree protocols [2]. In binary search protocols (BS for short), [12,11] the reader performs identification by recursively splitting the set of answering tags. Each tag has a counter initially set to zero. Only tags with counter set to zero can answer at the reader's queries. After each tag transmission, the reader notifies the outcome of the query: collision, identification, or no-answer. When tag collision occurs, each tag with counter set to zero adds a random binary number to its counter. The other tags increase by one their counters. In such a way, the set of responding tags is split into two subsets. After a no-collision transmission, all tags

Table 5
Pseudo-code of QT protocol
Reader procedure:
1. $prefix$ = empty;
2. *query*($prefix$,0);
3. *query*($prefix$,1);

query(char[] $prefix$,char c):
4. $prefix+ = c$;
5. broadcast($prefix$);
6. receiveAnswers;
7. **if** (channelStatus= 1) **then** *tagIdentification()*;
8. **else if** (channelStatus> 1) **then**
9. *query*($prefix$,0);
10. *query*($prefix$,1);

Tag procedure:
11. $identified = false$;
12. **while** (**not** $identified$) **do**
13. receive($prefix$);
14. $offset = prefix.length$;
15. **if** ($prefix = myID[0...offset-1]$) **then**
16. sendAnswer;
17. **if** received$IDrequest$ **then** send $myID$;
18. $identified = true$;

Table 6
Pseudo-code of QTI protocol
Reader procedure:
From step 1 to step 10 of Table 5
11. **else if** (channelStatus = 0) **then**
12. $prefix-$ =lastChar;
13. $prefix+$ =!c;*
14. *queryImp*($prefix$,0);
15. *queryImp*($prefix$,1);

* !$char$ is such that: if char=0, then !char=1, and vice versa.

Table 7
Pseudo-code of QTAA protocol
Reader procedure:
1. $prefix$ = empty;
2. *query*($prefix$,0,0); *query*($prefix$,0,1);
3. *query*($prefix$,1,0); *query*($prefix$,1,1);

query(char[] $prefix$,char c_1, char c_2):
4. $prefix+ = c_1 + c_2$;
5. broadcast($prefix$);
6. receiveAnswers;
7. **if** (channelStatus= 1) **then** *tagIdentification()*;
8. **else if** (channelStatus> 1) **then**
9. *query*($prefix$,0,0); *query*($prefix$,0,1);
10. *query*($prefix$,1,0); *query*($prefix$,1,1);

decrease their counters by one. Pseudo-code of BS protocol is reported in Table 1.

In query tree protocols (QT), the reader sends a prefix and tags having the ID matching the prefix answer. If there is a collision, the reader queries for one bit longer prefix until no collision occurs. Once a tag is identified, the reader starts a new round of queries with another prefix. We implemented and tested this basic version as described in Table 5. In [2], many improvements of QT protocol have been presented, for reducing its running time. Among them, the most important is the one that we call Query Tree Improved (QTI) [2], that avoids the queries that certainly will produce collisions. Assume that a query of prefix "q" results in a collision and the query of prefix "q0," results in a empty slot. Then, the reader skips the query prefix "q1" and performs directly the queries "q10" and "q11."

We implemented and tested this improved version of Query Tree protocol as described in Table 6, where only reader procedure is reported, since the tag procedure is the same as in QT protocol (see Table 5).

Another improvement proposed in [2], is the so called "aggressive advancement" (QTAA, for short), in which every internal node of the query tree has four sons: After the query of prefix "q" resulting in a collision, the reader does the following queries "q00," "q01," "q10," "q11." We implemented and tested also this version, as defined in Table 7.

Since tags do not need additional memory except the ID, query tree protocols have the advantage to be memoryless and for this they require low functional and less expensive tags. However, since they use prefixes, their performance is sensitive to the distribution of tag IDs which a reader have to identify. In Section 5, we show how ID's distribution affects the behaviour of query tree protocols.

3. Tree Slotted Aloha protocol

In this section, we present our proposal for tag identification which we call Tree Slotted Aloha (TSA) protocol. As we shall see in the next section, it performs better than the known protocols.

3.1. *The basic idea*

The basic idea of our TSA identification protocol, is to solve a collision as soon as it happens. In Framed Slotted Aloha protocols, two tags not colliding in a frame, can collide in the next frame. In our approach, the above situation is avoided, since when a collision occurs in a slot, only the tags generating such collision are queried in the next read cycle. As we shall see in the next section, this results in a better performance.

3.2. *The protocol*

We consider an RFID system consisting of a reader and a set of n passive tags. Each tag $t \in \{0, ..., n-1\}$ has a unique ID string $t_{id} \in \{0,1\}^k$, where k is the length of the ID strings. We assume that the reader does not know the exact number n of tags present in its communication range, but it can estimate it. For instance, a human operator can have a rough idea of the number of tags to be identified. Alternatively, it can be derived from the history of previous identification processes, by using statistical tools. Such an estimation l_0 is the starting frame size.

The protocol is performed in several tag reading cycles. A reading cycle consists of two steps: in the first step, the reader broadcasts a request for data by specifying the frame size l_i, in the second step each tag in the communication range of the reader, selects its response slot by generating a random number in the range $[1, ..., l_i]$ and transmits its ID in such a slot. The reader identifies a tag when it receives the tag ID without collisions. The behavior of the protocol follows a tree structure. The root node is the frame in the first reading cycle. Let l_0 be the size of such a frame, N_i being the number of tags transmitting their ID in slot i, with $i \leq l_0$, $N_i \geq 0$, $\sum_i N_i = n$. If $N_i \geq 2$, there is a collision in slot i. At the end of each reading cycle, if the reader realizes that collisions occurred, it starts a new reading cycle for each slot where there was a collision. This corresponds to adding new nodes in the tree, as sons of the node representing the above reading cycle, one son for each slot with collisions. The size of such new cycles is defined as described later, and the reader broadcasts such a size, together with the slot number of the previous frame (to address only the tags colliding in that slot), and the level of the tree. In each reading cycle, tags store the generated random number (i.e. the slot in which they transmitted their ID) and increase by one their own tree level counter, so that they know when are involved in later communications. Obviously, in each new reading cycle, collisions can occur. Each time a collision

Table 8
Reader and Tag procedures in TSA.

Reader procedure:
1. $level = 0$;
2. $l_0 = N_{EXP_0}$;
3. $s = -1$;
4. *collisionResolution(level,s,l_0).*

collisionResolution(level,slot,l_i):
5. broadcast($level, slot, l_i$);
6. **for** $s = 1$ **to** l_i **do**
7. receiveAnswers;
8. update c_0, c_1 and c_k;
9. **if** (channelStatus[s]$= 0$) **then** $c_0 + +$;
10. **if** (channelStatus[s]> 1) **then** $c_k + +$;
11. **if** (channelStatus[s]$= 1$) **then**
12. *tagIdentification()*;
13. $c_1 + +$;
14. $N_{EXP_i} = ChebyshevEstimation(l_i)$;
15. $l_{i+1} = \left\lfloor \frac{N_{EXP_i} - c_1}{c_k} \right\rfloor$;
16. **for** $s = 1$ **to** l_i **do**
17. **if** (channelStatus[s]> 1) **then**
collisionResolution(level + 1,s,l_{i+1});

Tag procedure:
18. $identified = false$;
19. $myLevel = 0$;
20. $previousValue = -1$;
21. **while** (**not** $identified$) **do**
22. receive($level, slot, l_i$);
23. **if** (($level = myLevel$) **and** ($previousValue = slot$)) **then**
24. $s =$ randomNumber mod l_i;
25. $myLevel + +$;
26. $previousValue = s$;
27. sendAnswer in slot s;
28. if received$IDrequest$ **then** send $myID$;
29. **else if** ($myLevel > level$) **then** $identified = true$;

is sensed, a new node (son of the node representing the previous cycle) is inserted in the tree, and another reading cycle is started. The whole process is recursively repeated until no collisions are detected in a cycle. The reader and tags procedures are shown in Table 8, and an example of protocol execution is shown in Figure 1.

Like in Framed Slotted Aloha, TSA is not memoryless, since each tag has to remember the random number generated in the previous cycle, and the level of the tree, as said before. Notice that the amount of memory needed by each tag is very small, namely few bits to remember the tree level and the slot number of the previous reading cycle. In Section 5, we show that TSA performs better than AFSA in terms of number of slots needed to identify

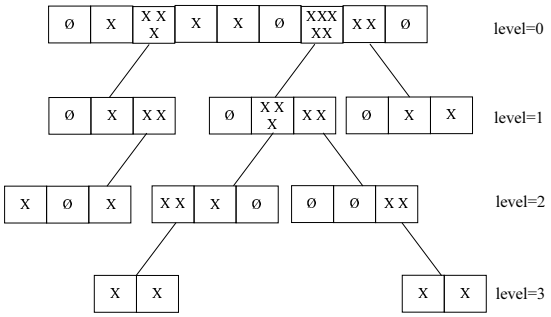

Fig. 1. Example of TSA protocol execution

all tags. As we shall see, our experimental outcomes are confirmed by the analytical result presented in [13] about the average size of a random hash tree, which is asymptotic to $2.3020238n$, where n is the number of uniformly distributed random entries to be placed in the tree. The tree built by TSA protocol is identical to such random hash tree, if we would know the exact number of tags. So, if this is the case, we can assert that the average number of slots needed to identify n tags is $2.3020238n$. In other words, if we define the system efficiency as the ratio between the number of tags and the number of slots needed to identify them like in [7], it results that the average system efficiency of TSA protocol is given by $n/2.3020238n = 0.4344$. In the same paper [13], the average height of a random hash tree is also shown. Such an height is proved to grow as $\lg_2 n$, in probability, with uniform distribution. So, in TSA the amount of bits needed to represent the tree level is equal to $\lg_2 \lg_2 n$. Such amount is actually very small: for instance, up to $n = 65536$, only $\lg_2 \lg_2 2^{16} = \lg_2 16 = 4$ bits are needed to represent the level of TSA tree (in addition to that ones needed to store the slot number).

3.3. *The estimation function*

The size of the frame in each reading cycle is computed by using a particular estimation function, similar to that used in [6], and defined as follows. At each reading cycle, we obtain a triple $< c_0, c_1, c_k >$ quantifying the empty slots, slots in which exactly one tag transmitted its ID, and slots with collisions, respectively. We use Chebyshev's inequality asserting that the outcome of a random experiment involving a random variable X, is most likely somewhere near the expected value of X. We use this property to compute the distance between the effective result $< c_0, c_1, c_k >$ and the expected result $< a_0, a_1, a_k >$

of a reading cycle. By minimizing such a distance, defined in equation (1), it is possible to estimate the number n of tags transmitting in such a cycle.

$$\epsilon(N, c_0, c_1, c_k) = \min_n \left| \begin{pmatrix} a_0^{N,n} \\ a_1^{N,n} \\ a_{\geq 2}^{N,n} \end{pmatrix} - \begin{pmatrix} c_0 \\ c_1 \\ c_k \end{pmatrix} \right| \quad (1)$$

The triple $< a_0, a_1, a_k >$ entries indicate the expected number of empty slots, slots filled with one tag, and slots with collision, respectively. N and n denote the frame size in the reading cycle and the number of tags, respectively. When the reader uses a frame size equal to N, and the number of responding tags is n, the expected value of the number of slots with r responding tags is given by

$$a_r^{N,n} = N \times \binom{n}{r} \left(\frac{1}{N}\right)^r \left(1 - \frac{1}{N}\right)^{n-r}.$$

We assume that the ϵ value in equation (1) is searched by varying n in the range $[c_1+2c_k, ..., 2(c_1+2c_k)]$, where $c_1 + 2c_k$ represents the minimum possible value of n. That is, since c_1 tags have been identified, and if there are c_k collisions, *at least* $2c_k$ tags collided [6]. We set the upper bound of the range equal to $2(c_1+2c_k)$, since by simulation we saw that there is no further accuracy in the estimation, if we set it to higher values. Details about this simulations outcome are not reported here since they are marginal for this paper. Our tag estimation function computes the frame size l_{i+1} of reading cycle $i+1$ as

$$l_{i+1} = \left\lfloor \frac{n^i - c_1^i}{c_k^i} \right\rfloor$$

where n^i, c_1^i and c_k^i are the estimation, the number of identified tags, and the number of slots with collisions related to reading cycle i, respectively. In our tag estimation function, we assume that if n^i is the number of transmitting tags in reading cycle i, the number of tags transmitting in reading cycle $i+1$ will be that one, minus the number of identified tags (c_1^i) in reading cycle i. Besides, since we assume uniform distribution of collisions, the number of unread tags in cycle $i+1$ will be the estimated number of colliding tags in cycle i divided by the number of slots with collision c_k^i. This function is applied for each reading cycle. Remember that cycle $i+1$ corresponds to one of the slots of cycle i in which a collision occurred (in terms of tree structure, it is one sons of node representing cycle i), and so it must be repeated c_k^i times.

3.4. *Complexity*

Tag identification protocols are usually compared according to two main performance metrics: *time complexity* and *bit complexity*. The former is the number of slots issued by the reader for identifying all tags. The latter is the amount of transmitted bits by the reader and/or by the tags, and represents the energy spent in communication. We analyze more in depth the time complexity, since this is the most used metric. Besides, as we shall show in the next section, by reducing the total number of collisions, our protocol reduces also the bit complexity (with respect to the Framed Slotted Aloha protocol). Notice that, since in TSA, tags store the number of the last slot in which they transmitted, it is possible to reduce the bit complexity by changing the protocol in this way: instead of always sending the whole serial ID number, tags can answer the reader by sending only one bit in each (micro)slot, and later the reader can query only the not colliding tags by asking for their ID. This holds if we assume that the reader can detect collisions by signal strength inspection.

4. Simulations Setting

We evaluated the behavior of the presented protocols by simulation. We implemented and tested also another version of TSA protocol, that we call Limited TSA (LTSA). It is equal to TSA except for a frame size adjustement, that is a power of 2. This may simplify the tag hardware. In particular, after computing the expected number of tags transmitting in the previous level reading cycle N_{EXP_i}, LTSA adjusts the frame size l_{i+1} according to the values given in Table 4.

We divided the simulation experiments in three parts. In the first one, we compared the performance of the protocols under the assumption of initial perfect knowledge of the number of tags to be identified. The purpose of this is to have a fair comparison of the protocols, since query tree based ones are not affected by the estimation (Figures 2-8). In the second part, we evaluated the variation of performance of TSA and AFSA with respect to the error in the initial estimation (Figures 9-12.) The third part was set up to get the performance of query tree based protocols when tag IDs are not uniformly distributed (Figures 13 and 14.)

Table 9
Computation of simulation metrics in **Scenario1**

	BS	DFSA	AFSA	TSA/LTSA	QT/QTI/QTAA
T	$+1$ *in Step 3.*	$+l_i$ *in Step 4.*	$+l_i$ *in Step 4.*	$+l_i$ *in Step 5.*	$+1$ *in Step 5.*
R	$+1$ *in Step 3.*	$+1$ *in Step 4.* $+1$ *in Step 8.*	$+1$ *in Step 4.* $+1$ *in Step 8.*	$+1$ *in Step 5.*	$+1$ *in Step 5.*
T_{bits}	$+BITS_{ID}$ *in Step 14.*	$+BITS_{ID}$ *in Step 18.*	$+BITS_{ID}$ *in Step 22.*	$+BITS_{ID}$ *in Step 27.*	$+BITS_{ID}$ - $offset$ *in Step 16.*
R_{bits}	$+8$ *in Step 3.* $+8$ *in Step 5.*	$+14$ *in Step 4.* $+8$ *in Step 8.*	$+19$ *in Step 4.* $+8$ *in Step 8.*	$+32$ *in Step 5.*	$+prefix.length$ *in Step 5.*

Table 10
Computation of simulation metrics in **Scenario2**

	BS	DFSA	AFSA	TSA/LTSA	QT/QTI/QTAA
T	$+1$ *in Step 3.*	$+l_i$ *in Step 4.*	$+l_i$ *in Step 4.*	$+l_i$ *in Step 5.*	$+1$ *in Step 5.*
R	$+1$ *in Step 3.* $+1$ *in Step 6.*	$+1$ *in Step 4.* $+1$ *in Step 8.*	$+1$ *in Step 4.* $+1$ *in Step 8.*	$+1$ *in Step 5.* $+1$ *in Step 12.*	$+1$ *in Step 5.* $+1$ *in Step 7.*
T_{bits}	$+8$ *in Step 14.* $+BITS_{ID}$*in Step 22.*	$+8$ *in Step 18.* $+BITS_{ID}$ *in Step 20.*	$+8$ *in Step 22.* $+BITS_{ID}$ *in Step 24.*	$+8$ *in Step 27.* $+BITS_{ID}$ *in Step 28.*	$+8$ *in Step 16.* $+BITS_{ID}$ *in Step 17.*
R_{bits}	$+8$ *in Step 3.* $+8$ *in Step 5.*	$+14$ *in Step 4.* $+8$ *in Step 8.*	$+19$ *in Step 4.* $+8$ *in Step 8.*	$+32$ *in Step 5.* $+8$ *in Step 12.*	$+prefix.length$ *in Step 5.* $+8$ *in Step 7.*

We consider two different system assumptions, resulting in the following scenarios:

– **Scenario1:** Tags transmit *always* their ID in BS, DFSA, AFSA, TSA and LTSA, while in QT, QTI and QTAA they transmit the suffix of their ID after receiving a given prefix. When no collision occurs, in BS, DFSA and AFSA, the reader identifies the tag and sends a message for stopping it; in TSA, LTSA, QT, QTI and QTAA, instead, this message is not necessary since tags realize their identification by the information included in following requests. In particular, in TSA and LTSA, a tag realizes its identification after receiving a query with a level value smaller than its own. In QT, QTI and QTAA, an identified tag never receives a prefix matching its ID. This scenario allows lower numbers of reader queries, but greater amounts of transmitted bits from tags.
– **Scenario2:** Tags transmit a prefixed byte of data, a kind of "Hello" packet. When no collision occurs, the reader issues a further query for getting the tag ID. This results in a greater numbers of reader queries, but in lower amounts of transmitted bits from tags.

In **Scenario1**, *sendAnswer* in protocol description of Tables 1 2, 3, 5, 6, 7 and 8 is equal to *sendMyID* (at most $BITS_{ID}$ bits transmitted), while in **Scenario2** it is equal to *sendHelloPacket* (8 bits transmitted). When no collision occurs, the *tagIdentification()* procedure consists of sending an acknowledge message (8 bits transmitted) in **Scenario1** (only in BS, DFSA, AFSA), while in **Scenario2** it consists of asking for the tag ID. Notice that, in this last case, we assumed that a time slot consists of two parts: the former for sending the answer, and the latter for eventual ID transmission.

We evaluated the following metrics:

(i) The number of transmission slots T; we will show the results for this metrics in terms of system efficiency, given as $S = N/T$.
(ii) The number of reader queries R.
(iii) The number of bits transmitted by tags T_{bits}.
(iv) The number of bits transmitted by the reader R_{bits}.

These values are computed for both scenarios as described in Tables 9 and 10 respectively. In particular, T is increased by l_i in Slotted Aloha protocols, and by 1 in query tree protocols. R is increased by 1 each time the reader transmits something. T_{bits} is increased by $BITS_{ID}$ in **Scenario1**, while in **Scenario2** is increased by 8 each time a tag answers a query, and once by $BITS_{ID}$ for identification. R_{bits} is increased by different amounts according to the transmitted data. We assumed the following values:

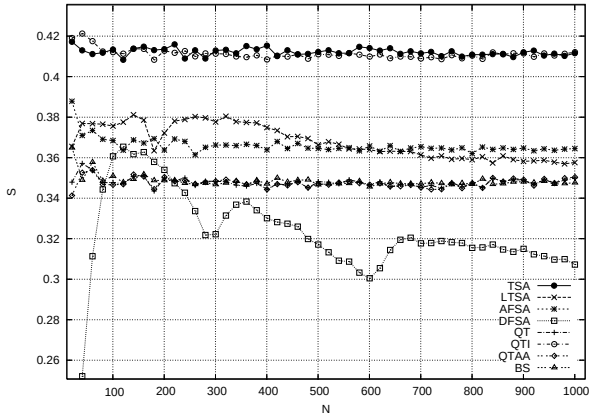

Fig. 2. S vs N, with $N_{EXP_0} = N$, $BITS_{ID} = 48$ and uniformly distributed IDs.

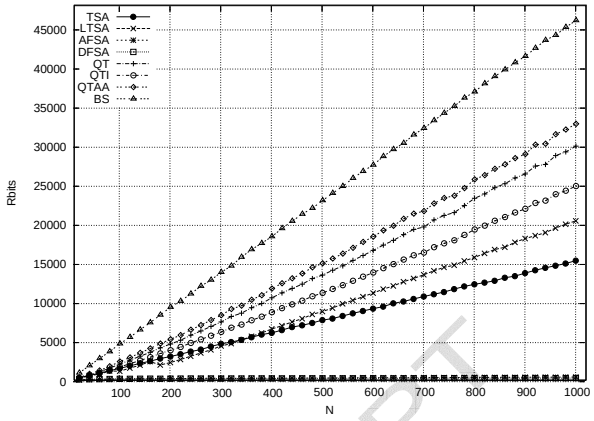

Fig. 3. R_{bits} vs N in **Scenario1**, with $N_{EXP_0} = N$, $BITS_{ID} = 48$ and uniformly distributed IDs.

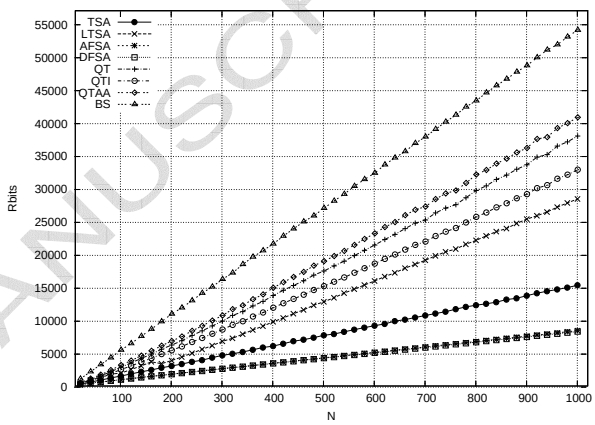

Fig. 4. R_{bits} vs N in **Scenario2**, with $N_{EXP_0} = N$, $BITS_{ID} = 48$ and uniformly distributed IDs.

14 bits for representing the frame length, 5 bits for the group number in AFSA, 4 bits for the tree level in TSA/LTSA, 8 bits for the *IDrequest* or the acknowledge message.

In the simulation experiments, we tuned the following parameters:

- The number of tags to be identified N: we tuned this parameter from 20 to 1000, by increasing it by 20 in each trial.
- The expected number of tags to be identified N_{EXP_0}: This value is used by Slotted Aloha based protocols for setting the initial frame size; we set this parameter to 2, 5%, 10%, 25%, 50%, 80%, 90% for under-estimation, and to 110%, 120%, 150%, 200%, 300%, 400%, 500% for over-estimation.
- The length of tags ID $BITS_{ID}$: We assigned to this parameter the values {48, 96, 128}.
- The type of ID distribution: The *uniform* one, and another distribution where groups of tag ID's are consecutive. In this last case, the maximum group size g was set equal to 10%, 20% or 50% of the number of tags N to be identified, and it was used in the ID generation in the following way. Together with the smallest ID of a group, which was uniformly generated, the size was set by generating another integer random number uniformly distributed in the range $[0..g-1]$. The group definition procedure checks also possible overlappings with already generated groups. In such a case, another smallest ID for a group was generated, until no overlapping was achieved.

Finally, each test ran 1000 times, and in the following section we show the obtained results.

5. Simulations results

We show here the results of the simulations of the protocols described above.

From Figure 2 we can see that the best performing protocols are QTI and TSA, in terms of total time needed to identify all tags, when IDs are uniformly distributed, and the ID length is 48 bits. For such a case, in Figures 3, 4, 5, 6, 7 and 8 the values of R_{bits}, T_{bits} and R are shown. T_{bits} represents the average number of bits transmitted by each tag. We observe that in **Scenario2** the numbers of transmitted bits by tags, T_{bits}, are quite small with respect to the same ones in **Scenario1**. On the contrary, the number of reader queries R (and of R_{bits}) is greater in **Scenario2**. Since in RFID systems it is more important to save energy in tags rather than in readers, then we can conclude that **Scenario2** is more appropriate in this kind of systems.

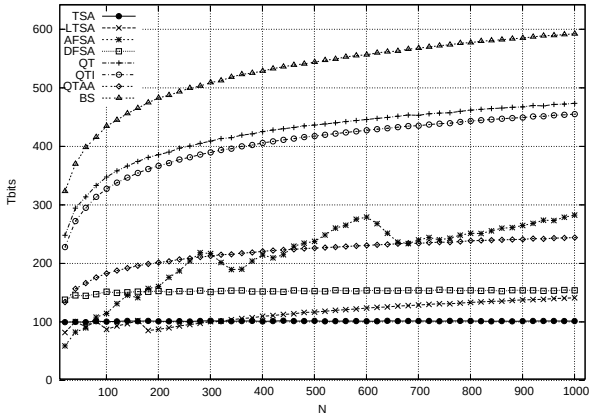

Fig. 5. T_{bits} vs N in **Scenario1**, with $N_{EXP_0} = N$, $BITS_{ID} = 48$ and uniformly distributed IDs.

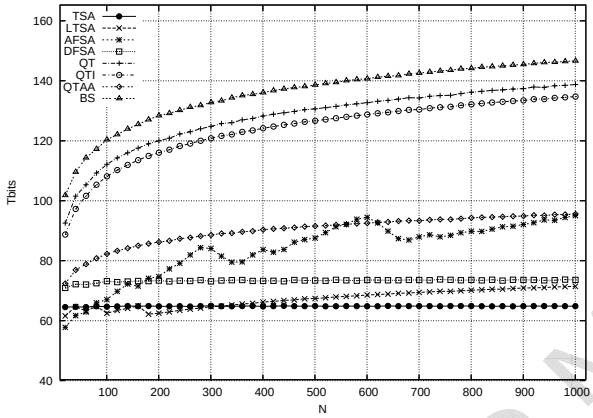

Fig. 6. T_{bits} vs N in **Scenario2**, with $N_{EXP_0} = N$, $BITS_{ID} = 48$ and uniformly distributed IDs.

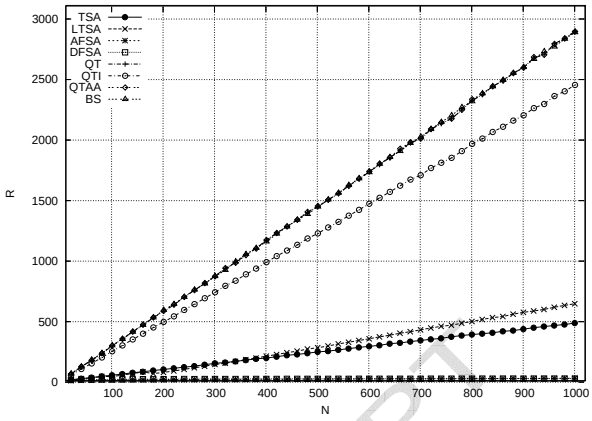

Fig. 7. R vs N in **Scenario1**, with $N_{EXP_0} = N$, $BITS_{ID} = 48$ and uniformly distributed IDs.

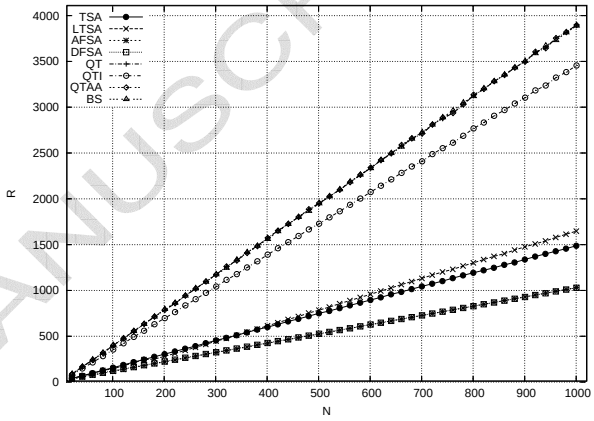

Fig. 8. R vs N in **Scenario2**, with $N_{EXP_0} = N$, $BITS_{ID} = 48$ and uniformly distributed IDs.

In Figures 9 and 10 we show the performance of TSA and AFSA with under-estimation; while in Figures 11 and 12 we show the same with over-estimation. The reason why TSA performs better when the initial frame length (equal to N_{EXP_0}) is a little smaller than the exact value of N, is the reduced number of empty slots. In fact, TSA is very efficient in solving the collisions, and so it is preferable to have a small number of collisions instead of empty slots. On the converse, AFSA is not efficient in collision solving, since it puts all the collided tags in one group, and so in later read cycles we can have collisions among tags that did not collide earlier.

In Figures 13 and 14 we evaluated the behavior of QT, QTI and QTAA protocols when the tag IDs are not uniformly distributed. From Figure 13 we deduce that the system efficiency increases by augmenting the maximum group size g. So, with small values of g, IDs are distributed in many small groups of consecutive IDs, and the reader is forced to issue many queries, since IDs of tags belonging to the same group are different only at the last bits. From Figure 14, we notice that this drawback degrades the performance when the ID length increases.

We conclude that in both scenarios, Tree Slotted Aloha protocol outperforms all the protocols presented in literature.

6. Conclusion

In this paper, we investigated the tag identification problem in RFID systems. Firstly, we surveyed the existing proposals for this problem. Then, we proposed a new probabilistic protocol based on a modified version of Slotted Aloha protocol, called Tree Slotted Aloha, to reduce the number of transmission collisions. All tags select a slot to transmit their ID by generating a random number. If there is

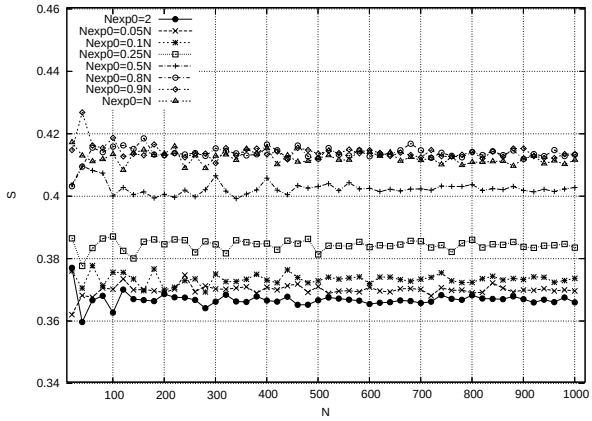

Fig. 9. TSA vs N_{EXP_0}, with under-estimation, $BITS_{ID} = 48$ and uniformly distributed IDs.

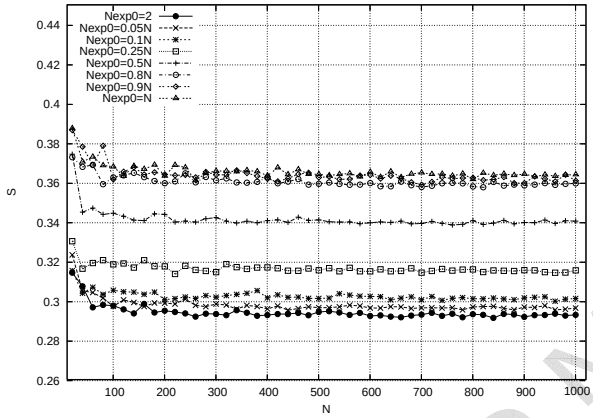

Fig. 10. AFSA vs N_{EXP_0}, with under-estimation, $BITS_{ID} = 48$ and uniformly distributed IDs.

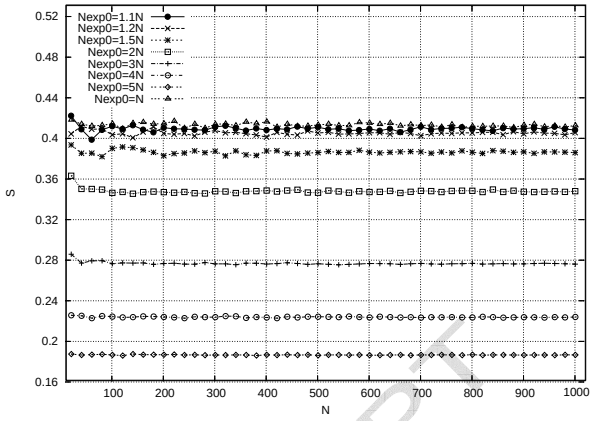

Fig. 11. TSA vs N_{EXP_0}, with over-estimation, $BITS_{ID} = 48$ and uniformly distributed IDs.

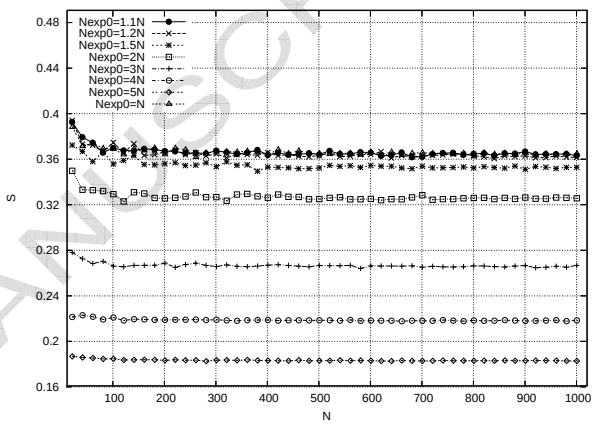

Fig. 12. AFSA vs N_{EXP_0}, with over-estimation, $BITS_{ID} = 48$ and uniformly distributed IDs.

a collision in a slot, the reader broadcasts the next identification request only to tags which collided in that slot.

Then, we presented the results of an intensive simulation experiment performed with the aim of thoroughly comparing the performance of the best RFID tag identification protocols. We first described in depth a simulation experiment set up to compare the performance of the above protocols, under several different metrics. Such a performance comparison is very useful when designing an RFID system, and, in our knowledge, was never carried out in the past. Furthermore, it is very useful for pointing out the best performing features of RFID tag identification protocols, and so for designing new and better protocols. According to the simulation experiment, our protocol Tree Slotted Aloha, outperforms all the other protocols proposed so far.

All the known protocols exhibit an overall system efficiency smaller than 50%. This is obviously not satisfactory, and so it would be very important to overcome such a performance limit.

Besides, uniform ID's distribution have always been assumed in the past. It would be very interesting to evaluate the performance of the protocols when such an assumption does not hold, and other distributions are more proper.

References

[1] M. L. Molle, G. C. Polyzos, Conflict resolution algorithms and their performance analysis, Technical report (1993).

[2] C. Law, K. Lee, K.-Y. Siu, Efficient memoryless protocol for tag identification (extended abstract), in: Proc. 4th International workshop on Discrete Algorithms and methods for mobile computing and communications (DIALM '00), New York, NY, USA, 2000, pp. 75–84.

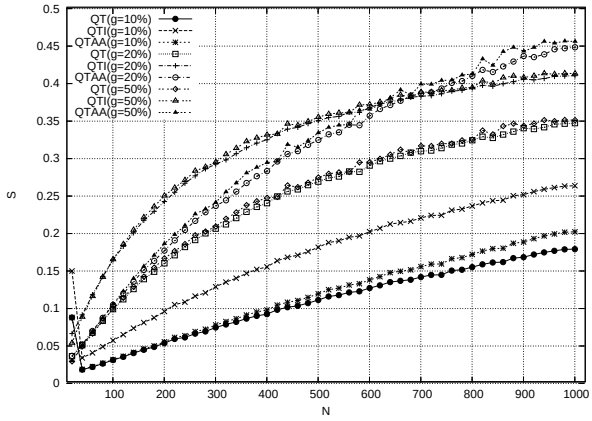

Fig. 13. S vs N, with $BITS_{ID} = 96$ and IDs distributed in groups.

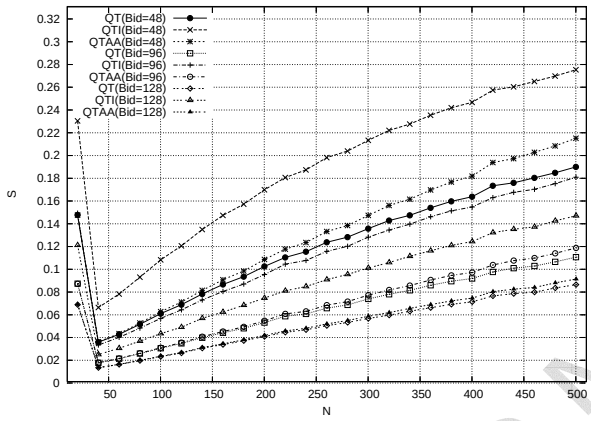

Fig. 14. S vs N, IDs distributed in groups with $g = 10\%$, and different values for B_{ID}.

[3] A. Micic, A. Nayak, D. Simplot-Ryl, I. Stojmenovic, A hybrid randomized protocol for RFID tag identification, in: First IEEE International Workshop on Next Generation Wireless Networks (WoNGeN '05), December 18-21, 2005.

[4] J. Myung, W. Lee, An adaptive memoryless tag anti-collision protocol for RFID networks, poster paper ats of 24th IEEE Annual Conference on Computer Communications (INFOCOM 2005), Miami, Florida, 2005.

[5] F. Zhou, C. Chen, D. Jin, C. Huang, H. Min, Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems, in: Proc. of International Symposium on Low Power Electronics and Design 2004 (ISLPED '04), New York, NY, USA, 2004, pp. 357–362.

[6] H. Vogt, Efficient object identification with passive RFID tags, in: Proceedings of International Conference on Pervasive Computing 2002, 2002, pp. 98–113.

[7] S.-R. Lee, S.-D. Joo, C.-W. Lee, An enhanced dynamic framed slotted aloha algorithm for RFID tag identification, in: Proceedings of Mobiquitous 2005, 2005, pp. 166–172.

[8] D. R. Hush, C. Wood, Analysis of tree algorithms for rfid arbitration, in: First IEEE International Workshop on Next Generation Wireless Networks (WoNGeN '05), August 1998, p. 107.

[9] F. C. Schoute, Dynamic frame length aloha, IEEE Transactions on Communications COM-31 (4) (1983) 565–568.

[10] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification, 2nd Edition, John Wiley & Sons, 2003.

[11] M. A.-I. Center, Draft protocol specification for a 900 MHz class 0 radio frequency identification tag, http://www.epcglobalinc.org (February, 23^{rd}, 2003).

[12] J. I. Capetanakis, Tree algorithms for packet broadcast channels, IEEE Transactions on Information Theory IT-25 (1979) 505–515.

[13] L. Devroye, The height and size of random hash trees and random pebbled hash trees, SIAM Journal on Computing 28 (4) (1999) 1215–1224.