5

MAC Protocols for RFID Systems

Marco Baldi and Ennio Gambi
Università Politecnica delle Marche
Italy

1. Introduction

Radio Frequency Identification (RFID) has become widespread thanks to its important advantages over traditional identification technologies, like barcodes. Compared to this technology, RFID is, in fact, able to cover larger distance and does not require line-of-sight, that represent important improvements in the considered field of application.
An RFID system is formed by a set of n tags, that represent clients in the identification process, and a tag reader, that is the identification server. Each tag is associated with an identifier (ID), that must be transmitted to the reader upon receipt of a suitable identification query. Tags can be active, i.e. provided with an independent power supply (like a battery), or passive. In the latter case, tags depend on energy provided by the tag reader through its queries (Want, 2006). For this reason, active tags can have some data processing ability, while passive tags are limited to doing only elementary operations and replying to the reader's queries.
An important issue in Radio Frequency Identification systems concerns the reading of co-located tags, that must be managed through suitable Medium Access Control (MAC) protocols, specifically designed for low power devices. Efficiency of MAC protocols for RFID systems directly influences the time needed by the tag reader for completely identifying a set of co-located tags, so it has great impact on the whole system performance. A deep analysis of the MAC technologies adopted in RFID is the aim of this chapter.
We denote by $T_{tot}(n)$ the time needed by the tag reader for completely identifying a set of n tags. Under ideal conditions, it should be $T_{tot}(n) = nT_{pkt}$, where T_{pkt} is the time needed to transmit one identification packet. However, in order to ensure collision-free transmission to each tag, a MAC protocol is needed that, inevitably, produces a time waste with respect to the ideal condition. The time actually needed for identification depends on several factors, like the technology adopted, the working frequency, etc. So, RFID systems can exhibit very different reading rates, typically ranging between 100 Tag/s and 1000 Tag/s.
The most common MAC protocols used in RFID systems can be grouped into two classes: *deterministic protocols* and *stochastic protocols*. Deterministic protocols basically coincide with tree traversal algorithms: all RFID tags form a binary tree on the basis of their identifiers, and the reader explores the tree in a systematic way, by repeating queries based on bit masks. Randomness is only in the tree structure (due to the choice of the co-located tag set), while the algorithm execution is pre-determined.
Stochastic MAC protocols are instead based on the framed slotted Aloha (FSA) algorithm, that requires each tag to make a constrained pseudo-random choice of an integer number in order to reduce the probability of collisions.

Source: Radio Frequency Identification Fundamentals and Applications, Bringing Research to Practice, Book edited by: Cristina Turcu, ISBN 978-953-7619-73-2, pp. 278, February 2010, INTECH, Croatia, downloaded from SCIYO.COM

Both deterministic and stochastic MAC protocols for RFID systems have some advantages and disadvantages. binary tree (BT) protocols have the advantage of very low tag complexity, since there is no need of implementing pseudo-random number generators within tags (except for singulation, when tags with the same identifier can be co-located). Furthermore, binary tree does not require the estimation of the number of co-located tags, that yields an additional effort before reading.
On the other hand, though needing a quite accurate estimation of the tags number, the FSA protocol can be more efficient than the binary tree protocol for large sets of co-located tags. Moreover, FSA does not need any particular bit coding of the tags identifiers and is intrinsically able to resolve ambiguity in the case of multiple tags with the same identifier.
Another important issue in RFID systems concerns power consumption, that influences the tags' lifetime. In fact, the same (possibly short) identification time can be achieved at the cost of many collisions or with a small number of collisions. In the latter case, power consumption is obviously minimized, with the effect of a prolonged tags' lifetime. In general terms, stochastic protocols are recognized to be more power efficient with respect to deterministic ones, due to the reduced number of collisions they produce (Namboodiri & Gao, 2007).
Both deterministic and stochastic MAC protocols have been included in standards and specifications for RFID systems. For example, EPCglobal Class 0 and Class 1 Generation 1 specifications adopt two different binary tree algorithms for the reading of co-located tags, and the same occurs in the first versions of the ISO 18000-6 standard for RFID systems (ISO/IEC, 2003). The more recent EPCglobal Class 1 Generation 2 specification introduced a new inventory technique based on the FSA protocol, that has also been included in the ISO 18000-6 Type C standard.
Several alternative solutions have been proposed in the literature for the implementation of anti-collision algorithms targeted to RFID applications (Cha & Kim, 2005), (Feng et al., 2006), (Lee et al., 2005), (Myung et al., 2006), (Park et al., 2007). For many practical and commercial applications, however, a fundamental requirement is that RFID tags must be very simple and inexpensive devices, often designed for single use. So, it is of main importance to develop low complexity anti-collision protocols able to solve the issues related to the shared medium while considering power and cost constraints.
This chapter studies both deterministic and stochastic MAC protocols for RFID systems proposed in standards, specifications and recent literature. Their principles are described and their performance is assessed and compared through theoretical and numerical arguments.

2. The binary tree protocol

The binary tree protocol is one of the most simple arbitration protocols, and it is based on the random splitting of the whole group of clients into two subgroups each time a collision occurs. This way, clients are progressively separated into smaller groups, until no more than a single client remains in each group. This is equivalent to put the clients on the nodes of a tree having maximum nodal degree 2.
In the binary tree protocol implementations used for medium access control in RFID systems, the binary splitting of tags into subgroups is based on the value of their IDs, that have fixed length and are supposed to be unique.
The tag reader is responsible for the management of the tree traversal procedure. The algorithm starts when the tag reader announces the tree traversal and transmits a binary

value. All tags having that value as the first bit of their ID reply by transmitting back the same value, while the other tags exit from the traversal and wait for another query (with different initial value).
The reader receives the transmitted values for the next bit, and it may or not detect a collision. In fact, it may happen that some tags transmit the same value, but the reader is not able to distinguish if transmission was from a single tag or more than one. In both cases, when a reply is received, the reader goes on with the binary splitting by choosing and transmitting a binary value for the next bit, until completion of the tag ID. Actually, according to the ISO 18000-6 standard, sending a null value corresponds to no transmission, that helps to reduce the power consumption.

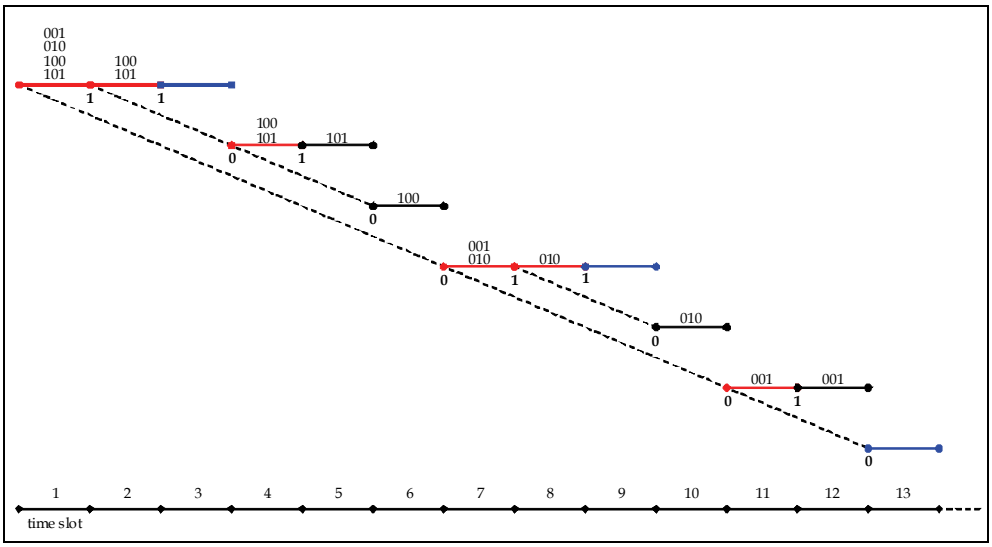


Fig. 1. Example of binary tree protocol with n = 4 tags.

An example of binary tree protocol used for reading co-located RFID tags is reported in Fig. 1. We suppose there are n = 4 tags with the following identifiers: 001, 010, 100 and 101. The time axis is divided into time slots; each time slot begins when the tag reader makes a new query for the value of a bit. The first time slot begins with the announcement of a multiple read query by the tag reader. All tags reply when such command is issued. The identifiers of the tags responding to each query are reported in the time slot that follows the query.
After the first time slot, each tag replies to the query if and only if the next bit of its ID coincides with the queried value; otherwise, it goes to sleep mode. When more than one tag reply in the same time slot, a collision occurs and the corresponding time slot is marked in red in Fig. 1. When the bit queried is not the last one, the tag reader is not able to tell a single reply from a collision, and must continue with the traversal. So, single replies at all bits except the last are equivalent to collisions (this occurs for the tags with IDs 010 and 001 in the figure). When no tag replies to a query, we have an *idle* time slot, marked in blue in the figure. Successful identification is instead accomplished when the last bit is queried and only one tag replies. The corresponding time slots are marked in black in the figure and labelled with the corresponding tag identifier.

An important feature of the binary tree protocol exemplified in Fig. 1 is that queries that are interrupted are then restarted exactly from the same point. In other terms, the tag reader must be able to store the status of each branch of the tree. This way, each edge of the tree is travelled only once, and queries are never restarted from the beginning. This version of the algorithm is with memory, and it is opposite to *memoryless* versions, whose efficiency is reduced due to repeated queries within the traversal (Bo et al., 2006).

Due to its simplicity, the binary tree protocol is suitable for being modelled analytically, and theoretical arguments can be used to predict the value of its most important parameters, that are (Janssen & De Jong, 2000): the number of tree levels required for a random contender to have success, the total number of tree levels and the number of contention frames required to complete the algorithm. Other relevant parameters, as throughput and delay, can also be estimated through analytical modelling of the protocol (Cappelletti et al., 2006).

The total number of tree levels obviously depends on the number of clients n, due to the assumption that, at the lowest tree level, each node must be associated, at most, to one client. The mapping between clients and tree nodes is stochastic, so the allocation of the tree nodes is not optimal. The total number of tree levels (D_n) can be lower bounded by considering the optimal distribution of clients on tree nodes. This occurs when even size groups of tags are split into equal subgroups, while odd size groups are split into subgroups having sizes that differ by one. In this case, it is simple to observe that:

$$D_n \geq \left\lceil \log_2(n) \right\rceil + 1, \tag{1}$$

where function $\lceil \bullet \rceil$ gives the smallest integer greater than or equal to its argument. In practice, due to the statistic nature of collisions, the number of levels is usually higher. In (Janssen & De Jong, 2000) it is proved that, for large n, the average number of tree levels is

$$D_n \simeq 2\log_2(n). \tag{2}$$

Another important parameter to evaluate the time needed by the binary tree protocol to complete the identification and to be compared with other arbitration protocols is the total number of time slots as a function of n. In order to estimate it for finite values of n (not necessarily large), we can resort to some simple theoretical arguments (Park et al., 2007).

First, we consider that each collision generates two edges, corresponding to the two possible choices for the bit under analysis. So, the total number of time slots required by the binary tree (I_{tot}) to completely identifying a set of n tags is simply twice the number of collisions (C_{bin}), augmented by one (to consider the first time slot):

$$I_{tot}(n) = 2C_{bin}(n) + 1 \tag{3}$$

If we focus on the i-th level of the tree, the number of time slots is $m = 2^i$, and each tag must reply to a query in one of them. The probability that a tag does not reply in a time slot is:

$$p(m) = 1 - \frac{1}{m} \tag{4}$$

and the probability that the time slot is idle (that is, no tag replies to a query in that time slot) is $p(m)^n$. So, the average number of idle time slots at level i is:

$$Q(n,m) = m \cdot p(m)^n = m \cdot \left(1 - \frac{1}{m}\right)^n . \quad (5)$$

We have reported in Fig. 2 the value of $Q(n, m)$ expressed by (5), as a function of the tree level (i), for different values of the number of tags (n). As expected, the average number of idle time slots quickly converges to the total number of time slots, that is, 2^i.

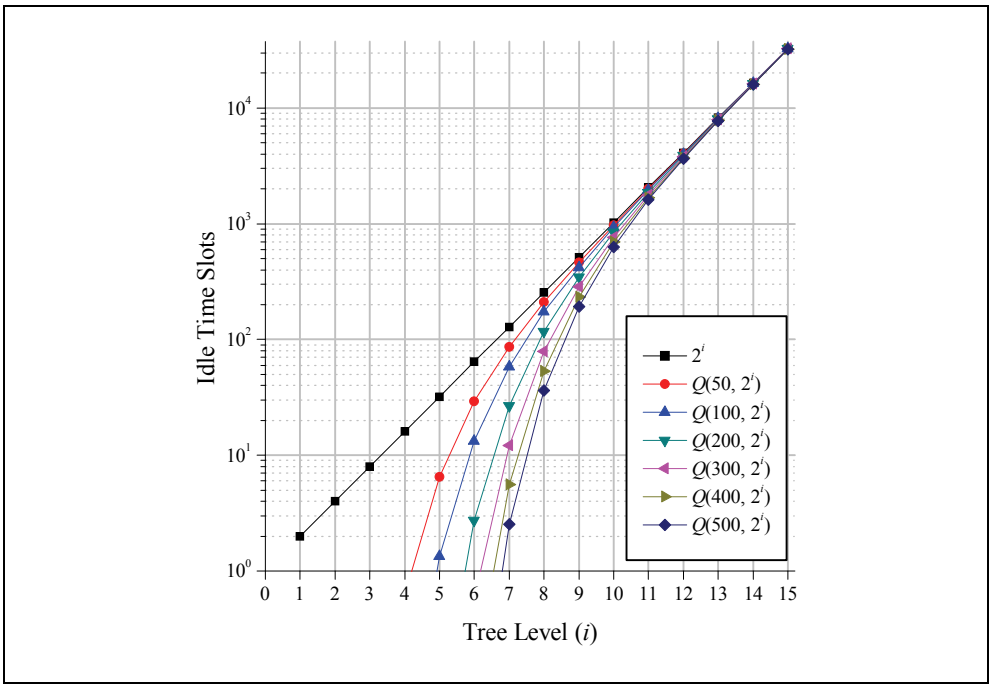


Fig. 2. Average number of idle time slots as a function of the tree level.

With similar arguments, we can consider that a time slot at level i is successful if it is used by a single tag to reply to a query. So, the average number of successful time slots at the tree level i can be estimated as follows:

$$S(n,m) = m \cdot n \cdot p(m)^{n-1}\left[1 - p(m)\right]. \quad (6)$$

Fig. 3 reports the average number of successful time slots as a function of the tree level, for the same choices of the number of tags, calculated by means of (6). In this case, it is immediate to observe that the number of successful time slots converges to the total number of tags n.

Starting from the expressions (5) and (6) for $Q(n, m)$ and $S(n, m)$, respectively, the average number of collisions at the tree level i can be estimated as the number of non-idle and non-successful time slots, i.e.:

$$C_{\text{bin}}(n,m) = m - Q(n,m) - S(n,m) . \quad (7)$$

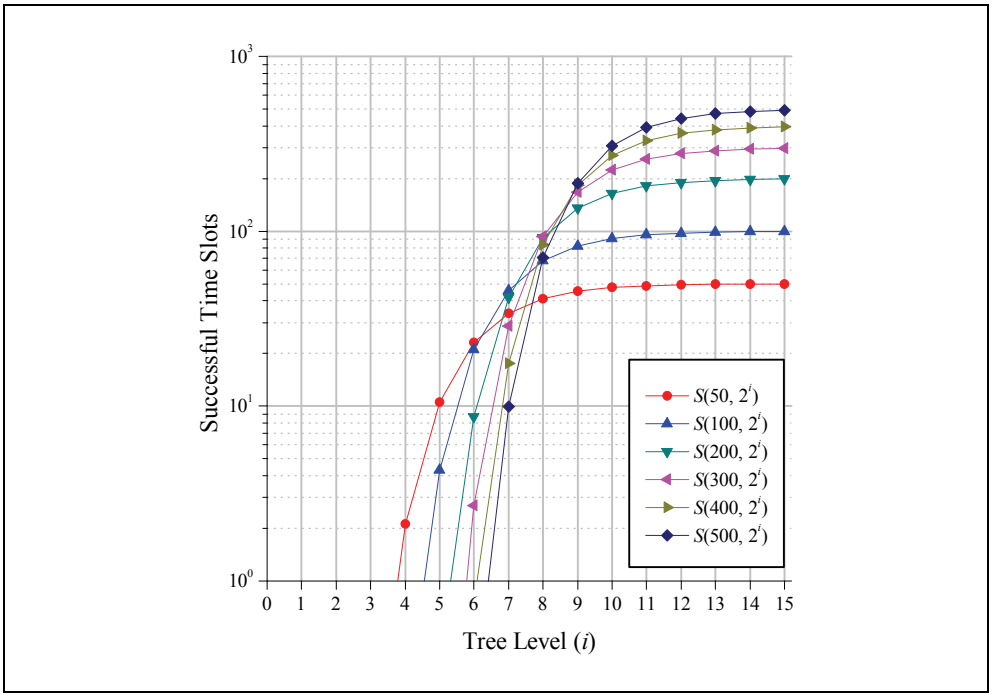


Fig. 3. Average number of successful time slots as a function of the tree level.

Based on these considerations, it follows that the average number of time slots with collisions rapidly goes to zero (a negative number of collisions obviously has no sense). For better evidence, the average number of collisions, expressed by (7), is reported in Fig. 4, for the same choices of n, as a function of the tree level. We observe that, for the initial tree levels, the number of time slots with collisions coincides with the total number of time slots (2^i). Then, the number of collisions becomes smaller than 2^i and, after reaching a maximum value, begins to decrease monotonically.

The total number of collisions in the binary tree protocol can be found by summing the average number of collisions at each tree level, that is:

$$C_{\text{bin}}(n) = \sum_{i=0}^{\infty} C_{\text{bin}}\left(n, 2^i\right). \tag{8}$$

The series surely converges because $C_{\text{bin}}(n, 2^i)$ becomes null for the values of i exceeding a given threshold. By substituting (5), (6) and (7), equation (8) can be rewritten as follows:

$$C_{\text{bin}}(n) = \sum_{i=0}^{\infty} m\left[1 + (n-1)\,p(m)^n - n \cdot p(m)^{n-1}\right]. \tag{9}$$

Expression (9) allows to determine analytically the total number of collisions and, through (3), we can then estimate the total number of time slots needed by the binary tree algorithm to completely identifying the set of n tags. As we will see in the following, such analytical estimation gives results that are very close to those of numerical simulations.

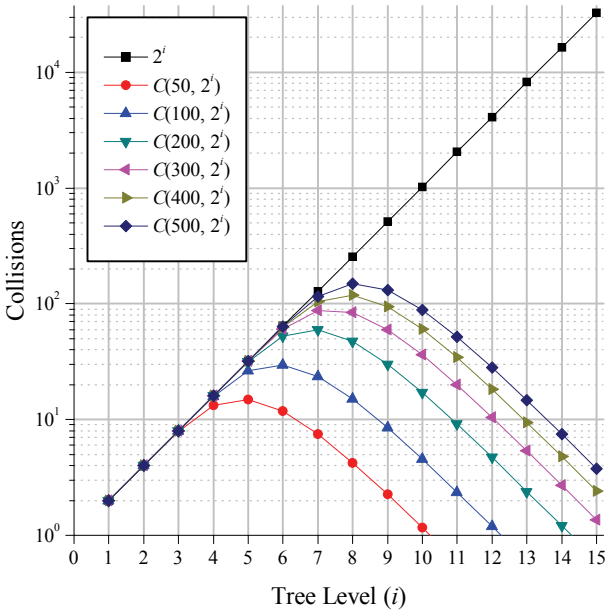


Fig. 4. Average number of time slots with collisions as a function of the tree level.

3. The framed slotted Aloha protocol

As we have seen in the previous section, the binary tree protocol for medium access control in RFID systems exploits binary splitting of the tags into subgroups on the basis of their identifiers, that are fixed and known a priori. For this reason, the binary tree and other similar deterministic protocols are opposed to stochastic protocols, mostly based on the framed slotted Aloha algorithm.

In slotted Aloha protocols, the time axis is divided into time slots. Each tag synchronizes its transmission with the beginning of a time slot, in such a way that concurrent transmissions collide completely. This is the main difference between slotted Aloha protocols and the pure Aloha one, in which instead the time axis is not discretized, so partial collisions can occur as well.

In the framed version of the slotted Aloha protocol, time slots are grouped into groups of L, and each group coincides with a frame. Each tag can transmit only once in each frame, and frames are repeated until the end of the identification procedure.

At the beginning of a frame, each tag randomly selects a time slot within the frame for transmitting its ID. The tag then transmits at the chosen time slot; if a collision occurs, colliding tags wait the end of the current frame and repeat the procedure in the following frame. The advantage of the introduction of frames is due to the limitation in the transmission rate imposed by the fact that each tag only transmits once in a frame. This allows to reduce the number of collisions in the initial phase of the protocol, when all tags try to communicate. This can result in a significant performance improvement with respect to slotted Aloha, on condition that the frame length is properly chosen.

In standard FSA, the frame length must be fixed a priori and is kept constant until completion of the algorithm. When the number of tags significantly exceeds the frame size, efficiency of the FSA protocol with fixed frames decreases. On the other hand, the algorithm efficiency could be kept high by adjusting the frame length on the basis of the number of active tags. In this case, however, such number must be estimated, that instead is not necessary in classic FSA.
A first solution to the problem of estimating the number of tags in the FSA protocol is to adopt dynamic versions of the FSA (Cha & Kim, 2005), (Lee et al., 2005). These approaches exploit the dependence of the probability of collision on the frame size and the number of tags. Such dependence can be expressed in analytical terms through theoretical arguments similar to those used in the previous section for the analysis of the binary tree protocol.
Let P_{idle}, P_{succ} and P_{coll} represent the probability that a time slot is idle, used for a successful transmission or occupied by a collision, respectively. Similarly to (5) and (6), we can express P_{idle} and P_{succ} as follows:

$$\begin{cases} P_{idle} = \left(1 - \frac{1}{L}\right)^n \\ P_{succ} = n\frac{1}{L}\left(1 - \frac{1}{L}\right)^{n-1} \end{cases}. \tag{10}$$

Starting from (10), P_{coll} can be calculated as:

$$P_{coll} = 1 - P_{idle} - P_{succ}, \tag{11}$$

so n can be estimated from the knowledge of L and the estimate of P_{coll}. However, it has been observed that the estimate of n so obtained can be inaccurate, since, for high values of the collision probability, small errors in the estimation of P_{coll} may produce significant deviations of the estimated n from its actual value (Park et al., 2007).
Another important result that can be derived from (10) is that the optimal frame size in the FSA algorithm exactly coincides with the number of tags n. So, FSA becomes less and less efficient when the gap between L and n increases.
For this reason, in Dynamic FSA (DFSA), the probability of collision is used to obtain an estimate of the number of tags that try to access the shared medium. This is repeated at each frame, in such a way that the frame length is dynamically adjusted on the basis of the actual number of contending tags. As anticipated, the estimated number of tags can be inaccurate. We will denote by α the ratio between the estimated number of tags and its exact value (thus, $\alpha = 1$ represents the ideal behaviour).

3.1 FSA with robust estimation and binary selection

An efficient approach for estimating the number of contending tags in the FSA protocol has been proposed in (Park et al., 2007), where the variant denoted as "FSA with Robust Estimation and Binary Selection", or EB-FSA, has been introduced.
The EB-FSA protocol begins with an estimation phase that has the purpose of estimating the number of tags n and to adjust the frame size L consequently. The estimation phase proposed in (Park et al., 2007) is robust, in the sense that it solves the issues due to high sensitivity of the estimated n to estimation errors on P_{coll}, in (11), for high values of the collision probability.

Estimation in EB-FSA starts by fixing an estimation frame size L_{est} and a target P_{coll} threshold ($P_{coll\text{-}th}$), that corresponds to a threshold number of tags (n_{th}) through (10) and (11). The tag reader estimates the value of n only when P_{coll} becomes smaller than $P_{coll\text{-}th}$, in such a way to reduce inaccuracy on the estimate of n (n_{est}). For this purpose, at each iteration of the estimation phase, the tag reader reduces the number of tags polled by a factor f_d, using a bit mask in the query frame.
Estimation of the tags number is made only when P_{coll} becomes smaller than $P_{coll\text{-}th}$, i.e., after a number of estimation frames equal to $i^*(n)$, that coincides with the smallest i such that $n/f_d^{i-1} < n_{th}$. In formula:

$$i^*(n) = \arg\max_{\substack{i \in \mathbb{N} \\ \frac{n}{f_d^{i-1}} < n_{th}}} \left(\frac{n}{f_d^{\,i-1}} \right). \tag{12}$$

Thus, in the EB-FSA protocol, the initial estimation phase requires $I_{est} = i^*(n)L_{est}$ time slots. After such initial phase, tags randomly choose an integer between 1 and n_{est} for transmission of their ID, and a frame with length n_{est} is transmitted. Differently from FSA, when a collision occurs, colliding tags do not wait for the next frame to retransmit their ID. On the contrary, a binary selection mechanism is implemented, that works as follows:

- non colliding tags increment their counters by 1;
- colliding tags randomly choose a binary value;
- colliding tags that chose 0 try retransmission at the next time slot;
- colliding tags that chose 1 try retransmission after one time slot;
- if another collision occurs, the procedure is repeated within the new set of colliding tags.

The binary selection mechanism avoids the need for subsequent frames, since each collision is necessarily solved through the splitting procedure. The protocol succeeds when all the tags have been identified, that is, after I_{iden} time slots. So, the total number of time slots needed by the EB-FSA protocol for completing its task is:

$$I_{tot} = I_{est} + I_{iden} = i^*(n)L_{est} + I_{iden}. \tag{13}$$

As it will be evident through numerical simulations, the EB-FSA approach can achieve a significant performance improvement with respect to FSA and DFSA.

4. Protocols comparison

In order to assess and compare the considered protocols for medium access control in RFID systems, we report in this section the results of numerical simulations that model different scenarios.
We are interested in estimating the time needed by the considered protocols to complete the identification phase, in order to compare their efficiency in arbitrating the channel use in groups of tags with different size. The actual identification speed depends on technology issues, so we refer to the number of time slots instead of real time.
We consider, as a starting point, the classic implementation of the framed slotted Aloha protocol, with fixed frame size. We consider two common values of frame size, that are L = 128 and L = 256, and estimate by simulation the total number of time slots needed to complete the identification procedure (I_{tot}). The results obtained are averaged over a number of simulations sufficiently high to ensure a satisfactory level of statistic confidence.

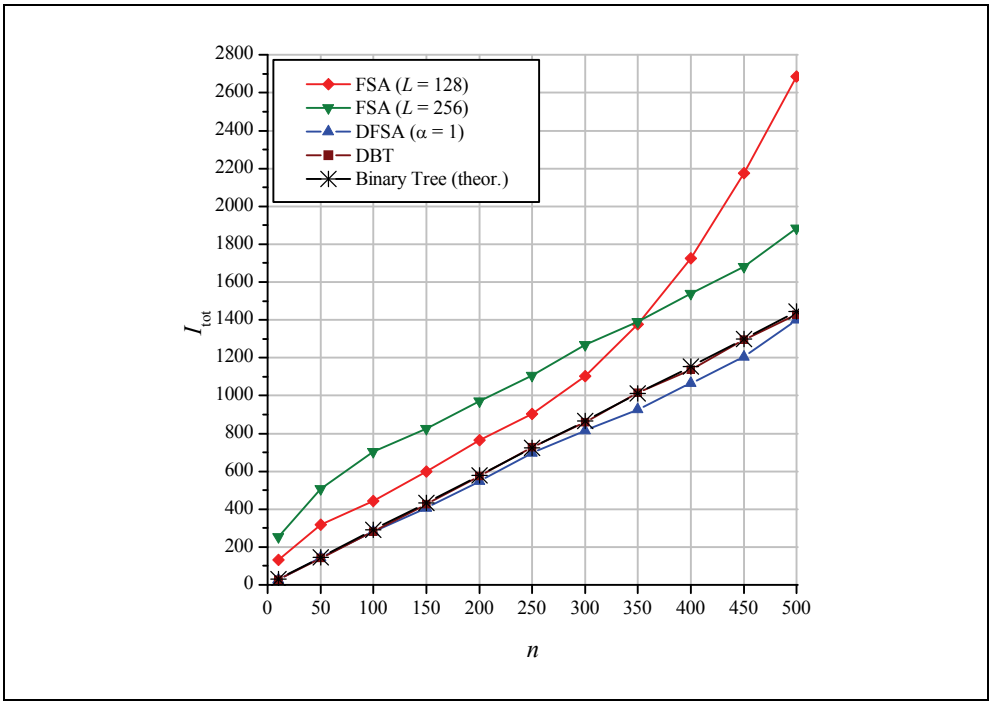


Fig. 5. Comparison of binary tree and framed slotted Aloha protocols.

The values of I_{tot}, so obtained, are reported in Fig. 5 as a function of the total number of tags (n). As we observe from the figure, the classic FSA protocol becomes less and less efficient for an increasing number of tags. The curve corresponding to L = 128 exhibits a parabolic behaviour starting from n on the order of 300. When L is increased up to 256, the protocol is less efficient for a small number of tags (n between 0 and 350), but its performance is improved for higher n. The parabolic behaviour of the curve for L = 256 is not apparent in the figure, since occurs for higher values of n with respect to the simulation scope.

In Fig. 5, classic FSA is compared with Dynamic FSA, in which the frame length is changed dynamically in such a way to coincide always with the number of contending tags. When the estimation of the number of tags is exact (α = 1), the DFSA protocol is able to significantly improve the performance of classic FSA. As we observe from the figure, the improvement is on the order of 200 time slots with respect to FSA with L = 128 and n up to 250. When n increases, the advantage of adopting DFSA instead of FSA becomes more and more relevant.

As an example of the binary tree algorithm, we consider a distributed binary tree (DBT) protocol that is self-adjusting, and that is directly managed by tags (Baldi et al., 2008). It recalls the classic version of the binary tree traversal, in which, when a collision occurs, each client randomly chooses a binary value. This is the same principle at the basis of the binary selection phase in the EB-FSA protocol. In DBT, the reader sends its query and all tags randomly choose a binary value. Tags that chose 0 try transmission at the first time slot available, while the others try transmission at the following time slot. If a collision occurs,

the same procedure used in EB-FSA is adopted. So, colliding tags randomly select another binary value, while all the others increment their counters by 1.

Such implementation of the BT protocol is independent of the bit coding of the tags IDs (that instead must be suitably chosen in the RFID standard BT protocol). Moreover, a fundamental role in RFID standard binary tree is played by the tag reader, that must perform the splitting procedure based on the tags IDs. On the contrary, in DBT the tags are able to manage the protocol autonomously, without the reader's queries. This way, the tag reader is not required to store the status of forked queries to avoid travelling each edge more than once. On the other hand, a drawback of DBT is that it requires tags to perform some processing (as for generation of pseudo-random binary values), so it could be difficult to implement with passive tags.

As we see from Fig. 5, performance of the DBT protocol is very close to the theoretical expectation, expressed by (9). DBT is able to improve significantly the performance of standard FSA and, for a number of tags up to 500, gives a moderate performance loss with respect to DFSA.

As a further assessment, we can compare the performance of the DBT protocol with that of EB-FSA. A simple example of application of the two protocols is reported in Fig. 6, where we consider the case of n = 5 tags. We observe from the figure that the only difference between the two protocols consists in the distribution of the initial values of the tag counters: in EB-FSA each tag must know the frame size (L = 5, in this case) and chooses its random value accordingly. In DBT, instead, each tag starts by choosing a binary value, without any knowledge on the frame size. However, even without needing any information on the number of tags, the DBT protocol is able to achieve the same performance as the EB-FSA, in the considered example, since both protocols complete identification in 8 time slots. Moreover, contrary to EB-FSA, DBT does not require any estimation phase before identification.

On the other hand, it should be observed that the DBT protocol produces a higher number of collisions with respect to EB-FSA, due to the fact that the initial values chosen by the tags are only binary. In EB-FSA, instead, tags can select initial values in the range between 1 and n_{est}. For these reasons, the DBT protocol is less efficient than EB-FSA under the power consumption viewpoint.

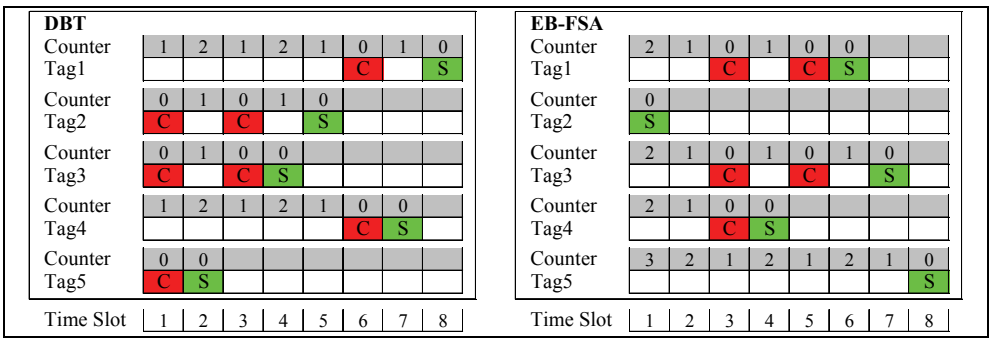


Fig. 6. Example of application of DBT and EB-FSA protocols.

When the number of tags increases, the advantage of having many possible initial random values in EB-FSA becomes more and more relevant, yielding a performance improvement

with respect to the DBT protocol. This is shown in Fig. 7, where DBT is compared with EB-FSA through numerical simulations, for a number of tags up to 500.

As we observe from the figure, under the hypothesis of perfect estimation of the number of tags, the EB-FSA protocol outperforms the DBT one. However, we can take into account the number of time slots needed by the initial estimation phase of EB-FSA, I_{est}, calculated on the basis of (12). The value of I_{est} has been found by considering, for the estimation phase, the same choice of the parameters proposed in (Park et al., 2007), that is, L_{est} = 64, $P_{coll\text{-}th}$ = 0.7 and f_d = 4. By considering the estimation phase, the performance gain achieved by EB-FSA becomes smaller and the two protocols have almost the same performance for a number of tags up to 300. So, the DBT protocol could still represent a valid choice, since it does not require the initial estimation phase, that has some drawbacks under the complexity viewpoint.

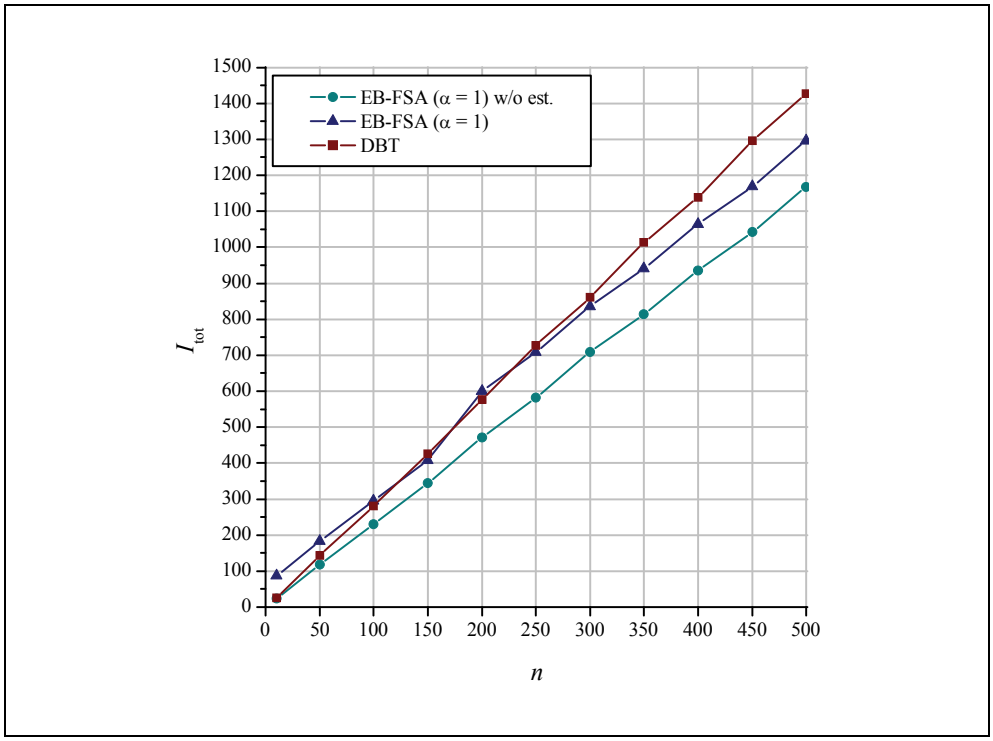


Fig. 7. Comparison of DBT and EB-FSA with perfect estimation (α = 1).

Another important aspect that must be taken into consideration is that, in the EB-FSA protocol, the estimation phase could be inaccurate, so the performance gain with respect to DBT could be further reduced.

We consider two different cases, in which we suppose that the reader estimates a number of tags equal to 0.5 and 1.5 times the actual number. The results of numerical simulations of the EB-FSA protocol with inaccurate estimation are reported in Fig. 8, where they are compared with those of the DBT algorithm.

We observe that the DBT protocol achieves almost the same performance as the EB-FSA in both the considered cases with inaccurate estimation. The overhead due to the initial estimation phase in EB-FSA has been also taken into account.
So, for a number of tags up to 500 (that is of interest for many applications), the DBT protocol (or, equivalently, the binary tree protocol based on tags IDs) is able to guarantee a rather good collision arbitration. Its performance compares with that of optimized stochastic algorithms, as the EB-FSA.
However, as we notice from the figure, the DBT curve has a higher slope with respect to those of EB-FSA and intersects them at $n \approx 300$. This confirms that, when the number of tags increases, the advantage of EB-FSA becomes more and more relevant.

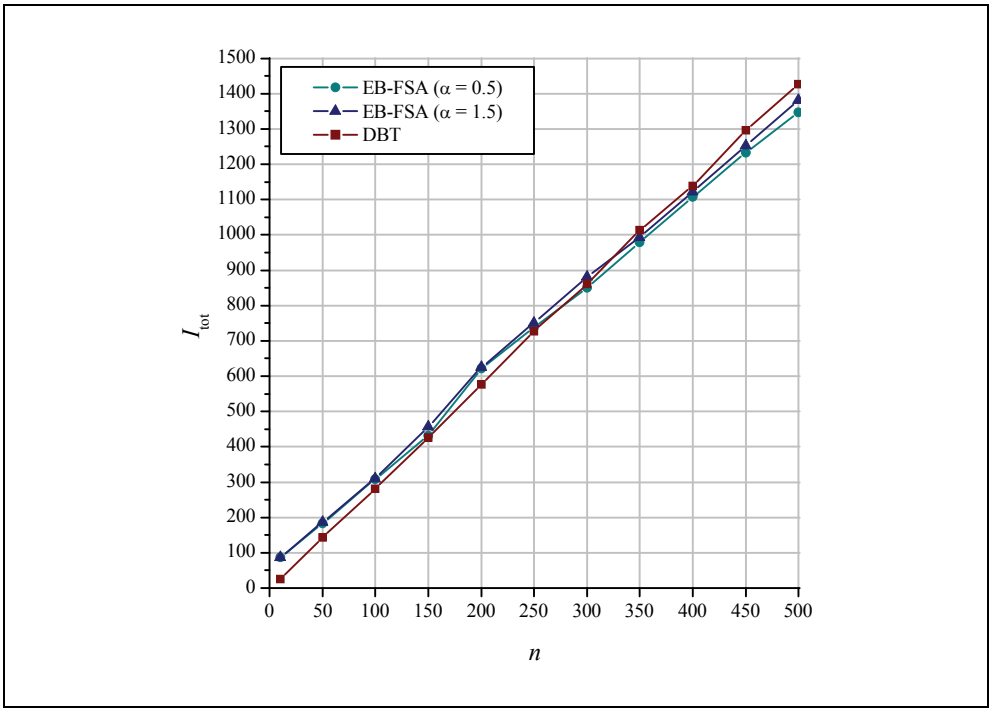


Fig. 8. Comparison of DBT and EB-FSA with inaccurate estimation (α = 0.5, 1.5).

5. Conclusion

A very common requirement in RFID systems is the reading of co-located tags, that is necessary when multiple tags are found simultaneously in the coverage area of the tag reader. Due to the low latency and low power constraints of these systems, the availability of efficient protocols able to arbitrate collisions and guarantee shared access to the transmission medium becomes of fundamental importance.
This chapter has described several anti-collisions protocols for RFID systems, that can be grouped in the two main categories of deterministic and stochastic protocols. We have seen that deterministic protocols, as the binary tree algorithm, can outperform classic stochastic

protocols as the framed slotted Aloha with fixed frame length. By considering more efficient versions of stochastic protocols, like the Dynamic FSA and the EB-FSA, the performance in terms of identification speed can be improved in a significant manner.
All these protocols, however, require estimation of the number of contending tags. So, the BT algorithm could still represent a good solution for resolving collisions without the need of tags estimation. We have seen that the BT protocol can also be implemented in a distributed manner, in such a way to be managed directly by tags and to be independent of the bit coding of the tags identifiers. This could represent an alternative to the RFID standard BT protocol (with centralized coordination) when intelligent tags are available.

6. References

Baldi, M., Morichetti, S. & Gambi, E. (2008). "A distributed binary tree protocol for medium access control in RFID systems", in *Proc. SoftCOM 2008*, Split, Dubrovnik, Croatia, 25-27 Sep. 2008, pp. 228-232.

Cappelletti, F., Ferrari, G. & Raheli, R. (2006). "A simple performance analysis of multiple access RFID networks based on the binary tree protocol", in *Proc. Intern. Symp. Commun. Control Signal Proc. (ISCCSP '06)*, Marrakech, Morocco, Mar. 2006.

Cha, J.-R. & Kim, J.-H. (2005). "Novel Anti-Collision Algorithms for Fast Object Identification in RFID Systems", in *Proc. ICPADS 2005*, Fukuoka, Japan, Jul. 2005, vol. 2, pp. 63–67.

Feng, B., Li, J.-T., Guo, J.-B. & Ding, Z.-H. (2006). "ID-Binary Tree Stack Anticollision Algorithm for RFID", in *Proc. IEEE ISCC '06*, Cagliari, Italy, Jun. 2006, pp. 207–212.

ISO/IEC 18000–6 (2003). "Information technology automatic identification and data capture techniques - radio frequency identification for item management air interface – part 6: parameters for air interface communications at 860-960 MHz", Nov. 2003.

Janssen, A. & De Jong, M. (2000). "Analysis of Contention Tree Algorithms", *IEEE Trans. Inform. Theory*, vol. 46, no. 6, pp. 2163–2172.

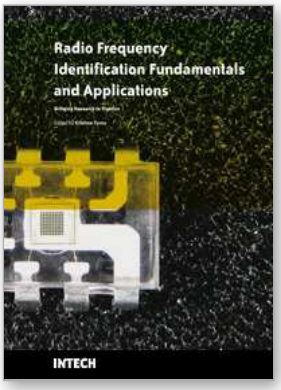
Lee, S.-R., Joo, S.-D. & Lee, C.-W. (2005). "An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification", in *Proc. MobiQuitous 2005*, San Diego, CA, Jul. 2005, pp. 166–172.

Myung, J., Lee, W. & Srivastava, J. (2006). "Adaptive Binary Splitting for Efficient RFID Tag Anti-Collision", *IEEE Commun. Lett.*, vol. 10, no. 3, pp. 144–146.

Namboodiri, V. & Gao, L. (2007). "Energy-Aware Tag Anti-Collision Protocols for RFID Systems", in *Proc. IEEE PerCom '07*, White Plains, NY, Mar. 2007, pp. 23–36

Park, J., Chung, M. Y. & Lee, T.-J. (2007). "Identification of RFID Tags in Framed-Slotted ALOHA with Robust Estimation and Binary Selection", *IEEE Commun. Lett.*, vol. 11, no. 5, pp. 452–454, May 2007.

Want, R. (2006). "An Introduction to RFID Technology", *IEEE Perv. Comput.*, vol. 5, no. 1, pp. 25-33.



Radio Frequency Identification Fundamentals and Applications
Bringing Research to Practice
Edited by Cristina Turcu

ISBN 978-953-7619-73-2
Hard cover, 278 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The number of different applications for RFID systems is increasing each year and various research directions have been developed to improve the performance of these systems. With this book InTech continues a series of publications dedicated to the latest research results in the RFID field, supporting the further development of RFID. One of the best ways of documenting within the domain of RFID technology is to analyze and learn from those who have trodden the RFID path. This book is a very rich collection of articles written by researchers, teachers, engineers, and professionals with a strong background in the RFID area.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Marco Baldi and Ennio Gambi (2010). MAC Protocols for RFID Systems, Radio Frequency Identification Fundamentals and Applications Bringing Research to Practice, Cristina Turcu (Ed.), ISBN: 978-953-7619-73-2, InTech, Available from: http://www.intechopen.com/books/radio-frequency-identification-fundamentals-and-applications-bringing-research-to-practice/mac-protocols-for-rfid-systems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821