A Simple Performance Analysis of Multiple Access RFID Networks Based on the Binary Tree Protocol

Fabio Cappelletti, Gianluigi Ferrari and Riccardo Raheli

Abstract—In this paper, we propose a simple performance analysis of radio frequency identification (RFID) networks, based on the binary tree medium access control (MAC) protocol. We evaluate various network performance metrics, such as throughput, delay and average number of packets needed to take a census of the RFID tags. In order to validate our analytical results, we develop a simple, yet accurate, simulator and we show that the predicted analytical performance is in good agreement with the simulation results. The significance of this work is due to the fact that RFID networks are becoming a widespread technology in industry and commerce.

I. INTRODUCTION

In the last years, radio frequency identification (RFID) systems have found an increasing application in several business areas, such as service, purchasing and distribution logistics, manufacturing, ticketing and animal identification [1]. RFID systems are currently available in the market, and developers need more and more to be able to optimize these systems for required specific applications. Therefore, it is very important to develop efficient tools to evaluate the performance of the systems under design.

In the literature, the study of RFID networks is somewhat immature, especially in terms of the evaluation of performance metrics such as throughput and delay. In [2], the author proposes an approach, based on the binary tree protocol (currently used in most RFID systems), to reduce the number of collisions in packet broadcast channels. In the same paper, a thorough analysis for the evaluation of the delay performance of the system is presented. In [3], the applicability of the binary tree protocol for arbitration in RFID networks is considered. In [4], a new collision arbitration protocol based on the well-known query tree protocol, characterized by the fact that the reader can remember tags which have collided, is proposed. Moreover, in [4] the authors compute the average number of time slots necessary to detect the entire tag population in the interrogator field and they also show that their new method significantly reduces the number of collisions for the successive reading operations. The goal of this paper is to propose a simple analysis of RFID network performance metrics, such as average delay and throughput. More precisely, the key idea is that of considering a deterministic approach, neglecting the stochastic aspect of the binary tree-based reading operation.

This simplifies tremendously the derivation of the analysis, still preserving its validity in an average sense. In order to validate our analytical results, we propose a reliable and scalable RFID network simulator. Our simulator is developed with OPNET 11.0 [5] and implements the binary tree collision arbitration protocol. The results obtained with our simulator are in good agreement with results recently appeared in the literature [6] and with those predicted by our analysis.

II. PRELIMINARIES ON BINARY TREE PROTOCOL

Due to the increasing application of RFID technology in the last years, several organizations have proposed various standards, such as ISO/IEC 18000 international standard [7], the AUTO ID center standard [8] and the GEN2 standard (which is the new generation ISO standard) [9]. The main difference between the above standards resides in the multiple access scheme. In fact, these protocols can be classified into two main categories: (i) the first one is based on the slotted Aloha medium access control (MAC) protocol [10] (ISO 18000 Type A), whereas (ii) the second group is based on the binary tree protocol (ISO 18000 Type B, GEN2 and AUTO ID) [2].

The binary tree protocol is a census protocol which efficiently reduces the problems of collisions [2], [3], [11]. The reader (or interrogator) illuminates the tags, which, through the small (but sufficient) absorbed energy, can transmit their identification (ID) information to the reader. Obviously, several tags can simultaneously transmit and, therefore, collide. In order to reduce efficiently the number of collisions, the key idea of the binary tree protocol consists in dividing a set of colliding tags into two subgroups: the tags belonging to one of the subgroups will not transmit until re-interrogated, whereas the tags belonging to the other subgroup will try to retransmit [7]. The same procedure can be used recursively to take a census on the latter group of tags. Eventually, only one tag will transmit, and the reader will correctly receive its ID information. The process which leads to correct identification of a single tag will be referred to as round. Taking a census of all tags in the network requires a number of rounds equal to the number of tags.

III. BINARY TREE PROTOCOL: A SIMPLE ANALYTICAL APPROACH

In this section, a novel and simple analytical approach for evaluating the performance of RFID networks is presented. Our analysis is based on a deterministic characterization of the binary tree protocol proposed in [7]. While our
The approach is suboptimal (since the stochastic nature of the collision-based resolution algorithm is neglected), it allows to derive simple performance bounds. We remark, however, that the developed OPNET simulator is realistic and takes into account the random nature of the binary tree-based query operation. The agreement between simulation and analytical results confirms the validity of our framework.

We first evaluate the total number of packets sent by the tags during a census of the entire RFID network. In order to do this, it is not necessary to take into account the number of *idle* time slots, i.e., time slots where no tag is transmitting. The idle time slots, however, have a strong impact on the delay required to take the census, and they will be taken into account later. We assume that if \( n \) tags transmit (and then collide) in a time slot, the number of tags that will retransmit their ID information in the consecutive time slot is \( n/2 \). This is the key difference between our analysis and more sophisticated statistical analyses, based on the approach originally proposed in [2]. Note that the validity of this simplifying assumption reduces when the binary tree characterizing the evolution of the tags becomes unbalanced, i.e., there are time epochs where the nodes do not split in equal subgroups after a collision. This phenomenon is exacerbated when the number of tags \( n \) is large, since there are many collisions and the probability of tree unbalancing increases significantly. The intuition behind this observation will be confirmed in the following, where the number of transmitted packets predicted by the analysis will be lower than that predicted by the simulator. However, the proposed analysis, on average, correct.

In order to make our approach clear, we consider the evolution of the binary tree protocol in a simple RFID network with 4 tags. The evolution of the binary tree, according to our simplifying assumption of regular subdivision (after each collision) of the not-yet-censed nodes, is shown in Fig. 1. Each node of the tree, depicted as a circle, represents a set of tags in the corresponding time slot (shown in the horizontal axis). The number above each circle corresponds to the number of tags associated with the node, whereas the number below each circle corresponds to the counter of the tags associated with the node. As explained in [7], only tags with their counters set to zero can transmit. For instance, after the collision in the first time slot, the 4 nodes divide into two subgroups of 2 nodes each: \( C_2 \) (tags which try to retransmit and set their counters to 0) and \( S_2 \) (nodes which remain idle and set their counters to 1). In order to compute the number of packets transmitted by the tags during the network census, we first consider a logical modification of the binary tree diagram used to describe the evolution of the RFID tag census operations. More precisely, we eliminate all *idle* circles: the binary tree shown in Fig. 1 is transformed into the binary tree shown in Fig. 2.

Suppose that there are \( n \) tags and refer to the compact version of the binary tree describing the evolution of the census operation, i.e., to the equivalent of Fig. 2. For the sake of analytical simplicity, we assume that \( n \) is a power of 2, i.e., \( n = 2^k, k \in \mathbb{N} \). In a scenario with a full compact binary tree, all nodes at each intermediate level of the tree (i.e., at depths lower than \( \log_2 n + 1 \)) correspond to the transmission of a packet for all \( n \) tags. Therefore, the total number of transmitted packets is

\[
P[n] = \sum_{i=0}^{\log_2 n} n = n(n \log_2 n + 1) \approx n \log_2 n \tag{1}
\]

where the last approximation holds for sufficiently large number of tags. In a generic scenario where \( n \) is not equal to a power of 2, the compact binary tree is incomplete. In this case, one can conclude that

\[
2^{\log_2 n} (\lfloor \log_2 n \rfloor + 1) < P[n] < 2^{\lfloor \log_2 n \rfloor + 1} (\lfloor \log_2 n \rfloor + 2). \tag{2}
\]

From (1), it follows that the average number of packets sent by a *single* tag during the network census is

\[
p_{\text{node}}[n] = \frac{P[n]}{n} = \log_2 n + 1 \approx \log_2 n.
\]

Network throughput can be defined as the ratio between the number of successfully transmitted packets (one per tag) and the total number of packets sent by the tags during the census:

\[
S[n] \triangleq \frac{n}{P[n]} = \frac{1}{\log_2 n + 1} \approx \frac{1}{\log_2 n}.
\]

The throughput per node can then be written as

\[
S_{\text{node}}[n] \triangleq \frac{S[n]}{n} = \frac{1}{n(\log_2 n + 1)} \approx \frac{1}{n \log_2 n}.
\]

At this point, we extend the previous analysis in order to compute the delay necessary to take a census of the tags.
The main difference with respect to the previous analysis is that in this case idle time slots, i.e., time slots where no tag transmits, have to be taken into account. With reference to the example with 4 tags considered before, note that, in computing (analyzing) the network delay, the binary tree in Fig. 1 cannot be compacted as in Fig. 2, since the entire evolution, including the idle time slots, is essential. The key point for the delay evaluation is then the computation of the number of idle time slots which follow a collision. In order to make an average analysis, we simply assume that the number of idle time slots is the average value associated with the number of tags which do not increment their counters. Let us denote by $n_c$ this number of tags—note that the number of tags associated with the immediately previous collision, i.e., a circle with a cross in the binary tree, is $2n_c$. The probability that the next time slot is idle is

$$p(\text{idle time slot} | n_c \text{ tags}) = \frac{1}{2^{n_c}}.$$ 

Therefore, denoting by $I$ the number of consecutive idle time slots, the average number of idle time slots after a collision among $n_c$ tags, denoted as $T[n_c]$, can be computed as follows:

$$T[n_c] = \sum_{i=0}^{\infty} i \left( \frac{1}{2^{n_c}} \right)^i = \frac{2^{-n_c}}{(1 - 2^{-n_c})^2}. \quad (3)$$

We are currently refining our analysis to correctly model the number of idle time slots between two consecutive transmissions.

The normalized total census delay, denoted as $T[n]$, can be expressed as follows:\footnote{Note that the normalized delay refers to the number of time slots. In order to derive the effective delay, one should consider $T[n] \times D_{pk}$, where $D_{pk}$ = $L/R_{th}$, $L$ is the packet length and $R_{th}$ is the transmission data rate.}

$$T[n] = n + C[n] + I[n]$$

where $n$ is the number of time slots where single tags correctly transmit their ID information, $C[n]$ is the number of time slots with collisions, and $I[n]$ is the number of idle time slots. Under the assumption that $n$ is a power of 2, the number of time slots with collisions is equal to the number of intermediate nodes in the compact binary tree:

$$C[n] = \sum_{i=0}^{\log_2 n - 1} 2^i = n.$$

The number of idle time slots $I[n]$ can be computed extending the approach previously used for evaluating $C[n]$. In fact, since the number of collision nodes in the tree at depth $k$ is $2^k$, the total number of idle time slots can be expressed as

$$I[n] = \sum_{k=1}^{\log_2 n} \sum_{i=1}^{2^k} T[n/2^k] = \sum_{k=1}^{\log_2 n} 2^k \frac{2^{-n/2^k}}{(1 - 2^{-n/2^k})^2}.$$ 

Therefore, the total delay $T[n]$ becomes

$$T[n] = n + \sum_{k=1}^{\log_2 n} 2^k \frac{2^{-n/2^k}}{(1 - 2^{-n/2^k})^2}. \quad (5)$$

In order to simplify expression (5), one can derive bounds on $I[n]$. In particular, the following lower and upper bounds for a generic term idle$_k$ in (4) can be obtained:

$$\text{idle}_k \geq \text{idle}_1 = \frac{2^{1-2^k}}{(1 - 2^{2^k})^2}, \quad \text{idle}_k \leq \text{idle}_{\log_2 n} = 2n.$$ 

From these bounds, it is possible to find the following lower and upper bounds for the number of idle time slots:

$$I[n] \geq \log_2 n \frac{2^{1-2}}{(1 - 2^{2})^2}, \quad (6)$$

$$I[n] \leq 2n \log_2 n. \quad (7)$$

Using (6) and (7), the total number of time slots $T[n]$ in (5) can be bounded as follows:

$$T[n] \geq 2n + \log_2 n \frac{2^{1-2}}{(1 - 2^{2})^2} \simeq 2n \pm T_{LB}[n]$$

$$T[n] \leq 2n[(\log_2 n) + 1] \simeq 2n \log_2 n \pm T_{UB}[n].$$

IV. PERFORMANCE EVALUATION

In Fig. 3, the network throughput and the throughput per node are shown as functions of the number of tags $n$. As one can see, there is good agreement between simulation and analytical results. In particular, the network throughput is $\Theta(1/\log_2 n)$, where the notation $\Theta(\cdot)$ means "on the order of". This confirms the fact that the binary tree protocol is a robust census protocol, in the sense that for increasing number of tags the throughput decreases much less than linearly.

In Fig. 4, the average time to census the entire tag population is shown as a function of the number of tags $n$. The meaning of the curves in Fig. 4 is explained below:

- The curve with filled circles shows the delay obtained by simulation: this is the realistic delay of the system.
• The curve with filled squares shows the delay $T[n]$ predicted by the analysis: as one can see comparing this curve with the simulation curve, the delay predicted by the analysis is lower than that predicted by the simulator.
• The dotted curve shows the lower bound on the delay, denoted as $T_{LB}[n] = 2n$.
• The dashed curve shows the upper bound on the delay, denoted as $T_{UB}[n] = 2n \log_2 n$.
• Motivated by the fact that upper and lower bounds are not close to each other, one can consider the arithmetic average of $T_{LB}[n]$ and $T_{UB}[n]$ (shown as dash-dotted curve):

$$T_{AA}[n] = \frac{T_{LB}[n] + T_{UB}[n]}{2} = n (1 + \log_2 n).$$

As one can see, the arithmetic average is slightly larger than the delay predicted by the simulator. The fact that the arithmetic average curve is not close to the simulation results depends on the fact that the terms $\{\text{idle}_k\}$ in (4) are not linearly increasing. Therefore, in order to obtain a more accurate approximation of the real delay behavior (predicted by the simulation), the upper and lower bounds $T_{UB}[n]$ and $T_{LB}[n]$ do not have to be combined with equal weights. By trial and error, we found that a simple and accurate expression for the delay is:

$$T_{heu}[n] = n \left( \frac{2 \log_2 n + 7}{3} \right).$$

The behavior of $T_{heu}[n]$ is shown in Fig. 4 as a solid line. As one can see, in the considered tag range ($n$ is between 1 and 100) the accuracy of $T_{heu}[n]$ is excellent. The average time to detect 100 tags is approximately equal to 1s, and this is consistent with the prediction in [6]—the slot time in our scenario is 0.16 ms. The obtained results lead to the conclusion that asymptotically, i.e., for sufficiently large values of $n$, the delay is $\Theta(n)$.

Finally, in Fig. 5 the average time required to detect a single tag is shown as a function of the number of tags $n$. Each curve shown in Fig. 5 is obtained by dividing the corresponding curve in Fig. 4 by the number of tags $n$. In particular, the delay curve predicted by the analysis, corresponding to the number of time slots $T_{node}[n] \triangleq T[n]/n$, is constant. This underlines the scalability of a binary tree-based multi-access network.

V. CONCLUSIONS

We have proposed a simple analytical approach to performance evaluation of a multi-access RFID networks based on the binary tree multi-access protocol. Neglecting the stochastic component of the evolution of the query process over a binary tree, we have been able to derive a simple analytical framework. The obtained results show that the network throughput scales as $\Theta(1/\log n)$ and the delay scales as $\Theta(n \log_2 n)$. The good agreement between analytical and simulation results proves the validity of our framework.

REFERENCES