8

Tree-Based Anti-Collision Protocols for RFID Tags

Petar Popovski

Department of Electronic Systems, Aalborg University

8.1 Introduction

Radio Frequency Identification (RFID) systems are increasingly and ubiquitously digitalizing the physical environment by attaching tiny tags to objects and people [1]. They are expected to support a large volume/diversity of applications, ranging from logistics, transport, retail services, access control as well as many yet-to-be-imagined applications belonging to the "Internet of Things" paradigm [2].

The two key components of an RFID system are tags and readers. A tag is a small microchip equipped with antenna which is attached to the object or a person. A tag can be passive (not using a battery), semi-active (using a battery for sensing/processing, but not for communication), or active (battery-powered tag with very low power consumption). The readers, also known as interrogators, are transceivers that can communicate with the tags, by reading information from tags or writing information to them.

Although a variety of RFID systems has been used for a considerable time [3], in recent years RFID technology has exhibited immense growth [4]. There are several reasons for such growth. Tags and readers are becoming less expensive, therefore their number and ubiquity increase. Clearly, inexpensive tags will be present in much larger numbers relative to readers. In particular, RFID systems with passive tags that operate in the UHF (Ultra-High Frequency) band are gaining importance due to the increased read range. The potential of such systems has increased dramatically in recent years due to the

RFID Systems: Research Trends and Challenges Edited by Miodrag Bolić, David Simplot-Ryl and Ivan Stojmenović
 ISBN: 978-0-470-74602-8

increased knowledge and applications of wireless networking. However, passive, batteryless tags pose unique challenges in the area of wireless networking due to the modest processing/storing capabilities of tags. Passive tags use the energy beamed from the reader to power their circuitry and also to transmit back to the reader using backscattering [3]. A tag can be attached to a sensor and act as a communication interface, thus creating a RFID sensor. One can easily conceive future systems comprising RFID tags and sensors that feature a combination of all types of tags; with passive tags far outnumbering active and semi-passive tags.

The communication mode in RFID systems can be described as follows: the reader sends a probe to a set of tags within its radio range. Based on the content of the probe, a tag decides whether to send its response by backscattering the power supplied by the reader. If more than one tag should respond to the probe, then multiple replies will simultaneously arrive at the reader, thus giving rise to the *tag collision* problem. If the replies of two or more tags collide, then there is very high probability that the signal will be corrupted and unreadable by the reader. Note that the tag collision problem occurs at the reader. If a tag is in the range of more than one reader, then a *reader collision* can occur at the tag [5]. This occurs when two or more readers transmit simultaneously, so that a tag that is in range of both readers cannot receive the probe sent by either readers (Figure 8.1).

Tag collision is resolved by running an anti-collision protocol (also called arbitration protocol or collision resolution protocol). The objective of an arbitration protocol is to make a transmission schedule for the tags, so that eventually each tag manages to send the reader a successful reply. Alternatively, instead of resolving the collision completely, the reader may need only partial resolution. For example, the reader may run a sequential decision process and gather data from the tags in order to carry out a statistical test of certain hypothesis and stop the arbitration process as soon as it has a sufficient set of responses. In general, the requirement of the arbitration protocol is determined by the higher-level task that is imposed on the reader network. Tag collision can be a particularly acute problem in UHF RFID systems, due to the larger reading range and thus the possibility of having larger number of tags in the reader's range.

Reader collision becomes a significant problem in environments with high reader density. Hence, the problem is to allocate communication resources (time, frequency) to the readers in order to minimize interference among them. For example, in Colorwave [6], the readers communicate with each other with the air interface that is also used to communicate with the tags and independently coordinate to minimize reader collisions using a

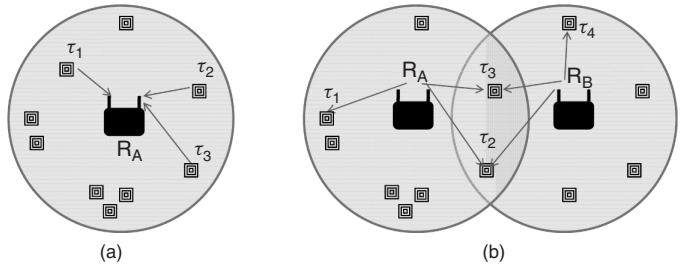

Figure 8.1 Illustration of (a) tag collision and (b) reader collision. Tag collision occurs at the reader R_A. The reader collision occurs at tags τ_2, τ_3, but not at τ_1, τ_4.

distributed time-division anti-collision algorithm. A comprehensive survey of the collision problems and collision resolution procedures in RFID systems is given in [7].

This chapter introduces the main ideas and perspectives of a class of anti-collision protocols termed *tree algorithms*. Tree protocols represent an important paradigm in the design of arbitration protocols for random access. These protocols have been introduced for a single channel with multiple access, where a single reader gathers replies from a population of tags that are in the range of that reader [8–10]. In essence, upon encountering collision, tree-based protocols recursively resolve the collision until all the tags initially involved in the collision have successfully sent their replies to the reader. Tree-based arbitration protocols represent one of the main approaches to resolve the tag collision problem in RFID systems [7, 11]. In recent years there have been a lot of studies that propose new variants of the tree algorithms [12–19]. The objective of this chapter is not to provide a detailed account of the different tree-based proposals and/or their quantitative comparison, but rather provide insight into the mechanisms used in tree-based approaches and possibilities to further optimize these algorithms in various scenarios.

This chapter is organized as follows. Section 8.2, where the principles of tree protocols are discussed in a generic setting, not strictly limited to RFID systems. This section first introduces several basic variants of the tree protocol, followed by techniques for improving the basic variants and finally an alternative arbitration framework is introduced which permits estimation of the tag population. Section 8.3 describes tree protocols that are used in specifications for UHF RFID systems. Section 8.4 presents practical issues that need to be considered, such as transmission errors and treatment of moving/late arriving tags. Section 8.5 discusses how tree protocols can be generalized to scenarios with multiple cooperative readers. The last section concludes the chapter.

8.2 Principles of Tree-Based Anti-Collision Protocols

This section introduces the principles and basic ideas used in tree protocols in a setting that is not necessarily limited to RFID systems. For example, we do not account for the computational/processing limitations of tags, error-prone communication, etc. The system model used in this section is idealized, but sufficient to introduce the mechanisms that are pertinent for understanding tree protocols. Relation to real-life systems and operation under non-idealized assumptions is given in Sections 8.3 and 8.4.

8.2.1 System Model

In this section we describe the context to present the basic ideas of tree-based anti-collision protocols. At the radio level, we assume that if a tag is within a distance D from the reader, then it can always receive probes from that reader without errors. Vice versa, if a single tag in the range of the reader transmits, then its packet is received successfully by the reader. We use the collision model for multiple access channel, which means that if two or more tags that are in the reader range transmit simultaneously, then the reader does not receive any of the transmitted packets. Note that these are rather strong assumptions for passive RFID systems, as many factors can cause transmission errors, such as fading, tag orientation, obstacles between the reader and the tag, etc. Nevertheless, this idealized MAC-layer model is sufficient to convey the concepts that are pertinent to the tree-based

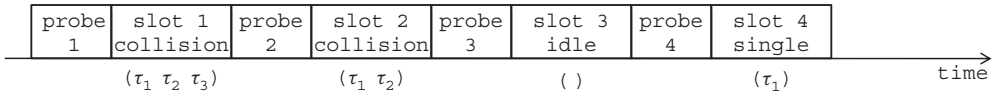

Figure 8.2 Time diagram that represents the channel over which the tag-reader arbitration and communication are done. Probes are sent by the reader, while in the other slots the reader receives. For example, in slot 1 tags τ_1, τ_2, τ_3 collide upon sending replies simultaneously.

protocols. In Section 8.4 we will relax the assumptions and discuss how various error types affect anti-collision protocols.

A tag sends a reply only if it is asked to do so by a probe sent by the reader. This is illustrated in Figure 8.2, where tags send replies in response to the probe. For example, upon receiving probe 1, tags τ_1, τ_2, τ_3 send their replies. By enabling replies from certain subsets of tags, the reader arbitrates the collisions on the channel. Note that the duration of the packet sent by each tag is constant and equal to a slot, so that if two tags transmit simultaneously their packets are completely overlapped. The reader then piggybacks feedback to tags in the next probe. For example, in Figure 8.2, `probe 2` sent by the reader carries feedback that informs the tags of the outcome (collision) in `slot 1`. When k tags transmit in the same slot, then the interrogator perceives the channel in that slot as:

- *Idle* (I) if $k = 0$ that is no tag replies.
- *Successful reception* (S) or *tag resolution* if $k = 1$.
- *Collision* (C) if $k \geq 2$.

In order to introduce tree-based arbitration protocols, we will assume that each tag is capable of producing random bits when asked to do so by the reader. This assumption will be revised in Section 8.4, where we will also discuss how the bits from the binary vector that describes tag's ID can be used for arbitration.

In the absence of errors, the efficiency of the arbitration protocols is measured both in terms of time and number of messages. If there are n tags, then we are interested in the average time $\bar{T}_n$ that the protocol consumes to identify all the tags. Let T_n be a random variable that stands for the time consumed when a particular instance of the arbitration protocol is executed. From Figure 8.2 it can be seen that T_n has two components: (1) the time used to send the probes; and (2) the time slots used for transmission from tags. We will assume that the duration of a probe sent by the reader is zero, and only focus on the time slots consumed for backscattering. With zero-length probe, the time efficiency of an arbitration protocol is defined as $\eta_n = \frac{n}{\bar{T}_n}$, see [20]. Note that another measure that is traditionally used to assess the performance of tree protocols is the average number of messages $\bar{M}_n$ sent during the arbitration process. This would be relevant if active tags were considered, as in that case each transmitted message consumes energy from the tag. On the other hand, when dealing with passive tags, $\bar{M}_n$ is rather irrelevant, as the tag energy is supplied by the reader and the average energy spent by the reader is linearly proportional to the average duration of the arbitration process.

8.2.2 Basic Tree Protocols

Tree-based protocols (also known as *splitting-tree protocols* or *tree-walking protocols*) have emerged as a solution to the multiple access problem over a shared medium. Several research groups have invented practically the same approach rather simultaneously [8–10]. The motivation for proposing these algorithms can be explained through a simple example with a single reader and two tags. The reader sends a probe requesting replies from the tags. Given that the reader does not know the tags' ID, this probe solicits an answer from unspecified tag and hence both tags are entitled to send a reply. After observing collision, the reader knows that there are at least two tags that have sent a reply. However, the number and the ID of these tags are unknown to the reader, giving rise to the symmetry problem, that is, all tags involved in the collision appear the same to the reader. In order to break this symmetry, the reader uses randomization. This means that after the collision, both tags toss a fair coin to generate a 0 or 1 value and in the next probe the reader asks only the tag that obtained 0 to send a reply. This procedure is repeated until the two tags have obtained different values for the random bits.

Figure 8.3 shows, through an example, how this basic idea can be generalized to more than two tags. In the example, the total number of tags τ_i is equal to $n = 8$. We define collision multiplicity (or conflict multiplicity) to be the number of tags that transmit when a collision is observed. The node labeled s_j refers to the outcome in the j-th slot. For example, the collision in the first slot s_1 has a multiplicity of eight, because all the tags sent their replies when responding to the initial probe. The level of a tree node is the path length from that node to the root of the tree. Each tree node is uniquely associated with a string called *address*. The address of a tree node is determined by the tossing outcomes for the tags belonging to that node. Using the tree representation, we can say that an

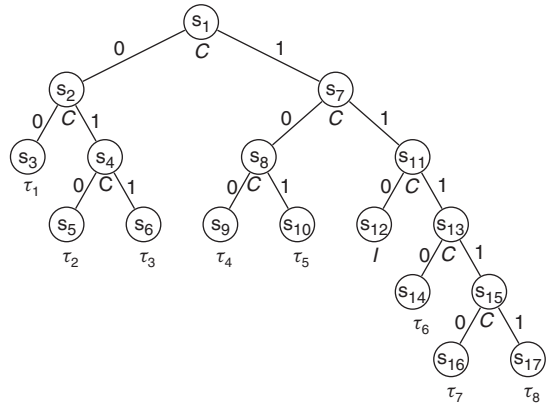

Figure 8.3 An instance of the binary tree algorithm for collision multiplicity of $N = 8$. Each vertex represents a slot that can be in one of the three states: Idle (I), Single (S) or Collision (C). For channel state "S," τ_i denotes the resolved or singulated tag.

address (a node) is enabled in slot s_i if the tags that belong to that node are allowed to transmit in s_i. At slot s_1 the tree root (level zero) is enabled, all eight tags transmit and the reader observes a collision. The address of the root node is empty, denoted by ϵ. Each tag tosses a fair coin and, in this instance of tree protocol, τ_1, τ_2 and τ_3 obtained value 0. At slot s_2, the left descendant of the root node is enabled and τ_1, τ_2, τ_3 are enabled to transmit. Note that the node enabled at slot s_2 is at level one of the tree and has address 0. The tags $\tau_4 - \tau_8$ that obtain coin value 1 after slot s_1 belong to the other node at level one, with the address 1. Having collision in slot s_2, the tags τ_1, τ_2 and τ_3 again toss fair coins. Only τ_1 obtained 0, while τ_2 and τ_3 obtained 1. Thus, when the node with address 00 (at level two of the tree) is enabled, only τ_1 sends a reply that is correctly received by the reader. Having transmission from a single tag, at slot s_4 another node at the same level is enabled, with address 01. Based on this explanation, the reader can go through the other steps of the example in Figure 8.3. Figure 8.5 shows the timing diagram of the tree algorithm from the example on Figure 8.3. It can be seen that the reader recursively resolves collisions until it obtains single (successful) answer from each tag.

Figure 8.4 shows a possible implementation of the basic variant of binary tree protocol by using a pseudocode. The code for the algorithm run at the reader represents one complete tree traversal. By setting **a** = *empty*, the reader can restart the traversal from the root of the tree. The pseudocode at the tag assumes that each probe carries the complete address. Note that the original tree algorithms proposed by [10] run in a completely distributed manner, without centralized probes, by using stack-based implementation.

```
a = empty; end=no;
while(end == no)
    transmit probe with address a;
    L = length(a);
// if a is not empty, then a = (a_1 a_2 ··· a_L)
    receive tag-reply;
    if tag-reply = IDLE
        transmit feedback;
    if tag-reply == COLLISION
        if L == 0
            a = (0);
        else
            a = (a_1 a_2 ··· a_L 0)
    else
        if a == empty
            end = yes;
        else
            while aL == 1
                delete a_L;
                L = L-1;
            if L == 0
                end = yes;
            else
                a_L = 1;
```

(a) The algorithm run at the reader

```
a = empty ;end = no;
while (end == no)
    receive pr; %this is the address in the probe
    if a == pr;
        transmit;
        get feedback at the end of the slot;
        % current address is a = (a_1 a_2 ... a_L)
        if(collision)
            set b = randombit;
            set a = (a_1 ... a_L b);
        else
        if(single)
            set end = yes;
```

(b) The algorithm run at a tag

Figure 8.4 Pseudocode of the algorithms run at the reader and at a tag for the basic variant of binary tree protocol.

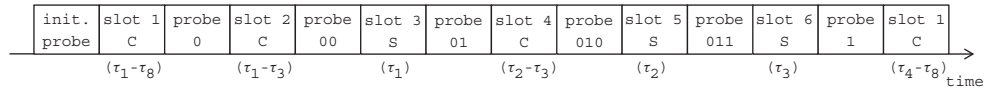

Figure 8.5 Timing diagram for part of the protocol execution described in Figure 8.3. The notation $(\tau_i - \tau_j)$ below a slot denotes the set of tags $\tau_i, \tau_{i+1}, \tau_{i+2} \ldots \tau_j$ that transmits in that slot.

Tree algorithms belong to the class of collision resolution protocols, rather than collision avoidance protocols, where a representative of the latter are the protocols that use random backoff, such as the MAC protocol for IEEE 802.11 [21]. When there are no other errors except collisions, the tree algorithm is guaranteed to collect a reply from each tag that is in the range of the reader. An interesting dividend is that the algorithm creates ad hoc addresses for tags. Namely, after the collision resolution, the reader might need to communicate with a particular tag. Instead of using the full address of that tag, the reader can assign a short address to the tag, and this address is represented by the address of the tree node in which this tag has been successfully resolved or singulated. For example, in Figure 8.3 the short ad hoc address of the tag τ_3 is 011.

8.2.3 Improvements to the Basic Tree Protocol

In this section we describe several ideas that improve and generalize the basic tree protocol, described in the previous section.

At first, note that the described tree algorithm constructs a tree by binary branching as dictated by bit flipping. The branching can be generalized to M rather than two outcomes in the randomization process. Larger M decreases the probability of collision, but increases the probability of obtaining an idle slot. If the idle slot has the same duration as a single/colided slot, then it is clear that increasing M does not increase efficiency. However, an idle slot can have a shorter duration than a slot in which one or more tags are transmitting. This is because the reader, in principle, does not need to wait for the whole duration of the slot to detect that no tag has transmitted. Hence, the duration T_I of an idle slot can be lower than the duration of the slot with tag transmission, denoted by T_S. Such a model for collision resolution is termed a Carrier Sense Multiple Access (CSMA) model. If $T_I \ll T_S$, then an idle slot is very "cheap" in terms of duration and the tree protocol should be engineered in a way to produce idle slots with higher probability as compared to the probability of occurrence of collided slots. Nevertheless, in this text we will not specifically treat the CSMA model. Possible optimizations of the tree-based algorithms in CSMA setting are discussed in [22] and [18].

Massey in [23] and Tsybakov and Mikhailov in [10] proposed a simple way to improve the basic variant of the binary tree algorithm. They noticed that there are some tree nodes that certainly contain more than one tag and will thus certainly result in collision if the node is enabled. These nodes should be skipped during the traversal of the tree. For example, in Figure 8.3, after the collision in slot s_{11} and an idle slot s_{12}, it is clear that the enabling of the address 111 results in certain collision. Hence, after the idle slot s_{12}, the terminals belonging to node 111 toss a coin immediately and in slot s_{13} the enabled node is 1110. This algorithm will be referred to as Modified Binary Tree (MBT) algorithm.

However, as discussed in Section 8.4, this small optimization has a very high price if transmission errors occur with non-zero probability.

Early on, researchers observed that knowing the multiplicity (number) of tags involved in a collision can speed up the resolution of that collision. Capetanakis in [9] observed that binary tree algorithms are most efficient for conflicts of small multiplicity and applied this observation to devise a *dynamic tree algorithm* for scenarios with Poisson arrivals. The idea is to divide the initial set of n tags into smaller groups and then apply a tree protocol within each group. Multiplicity estimation to speed up tree protocols has been used in [24] and [20]. In both works, the proposal is a hybrid algorithm that consists of two phases. The first phase is devoted to the estimation of multiplicity. After obtaining an estimate $\hat{n}$, the second phase starts. The unresolved terminals are randomly split into approximately $\hat{n}$ groups and each group is resolved using the basic tree algorithm.

Note that both [24] and [20] have an explicit phase for estimating $\hat{n}$ and the accuracy of such an estimate is using algorithm-specific parameters that are chosen in advance. On the contrary, in [18] we have introduced a new framework for tree-based arbitration protocols where a running estimate $\hat{n}$ of the initial population is obtained, which becomes increasingly accurate as the collision resolution progresses. Before proceeding to the details of the framework from [18], we introduce the Clipped Binary Tree (CBT) algorithm, which is the main ingredient of tree algorithms with a running estimate.

CBT algorithm has been independently introduced by several authors in the context of random access for an infinite population of terminals that generates transmission requests with Poisson arrivals [25]. It is identical to the MBT algorithm except that it is stopped, that is, the tree is clipped whenever two consecutive successful transmissions follow a conflict. CBT resolves the batch conflict partially, since not necessarily all nodes of an initial batch are resolved during the execution of CBT. For example, referring to the tree in Figure 8.3, the first instance of the CBT algorithm starts in slot s_1 and terminates in slot s_6, resulting in three resolved tags. Now, the standard binary tree algorithm would continue to enable the node labeled with s_7, that is, all the tags that have flipped 1 after the initial collision. The key observation from [18] can be applied to this example as follows. After the termination of the first CBT algorithm, it is noted that 3 tags have flipped 0, so that the expected value of the number of tags that have initially flipped 1 is 3. This implies that the enabling of the node s_7, which belongs to the level 1 of the tree, will highly likely result in collision and therefore the next probe should directly enable a node that has a higher level in the tree. For example, if the node with address 10, from level 2, is enabled after the termination of the initial CBT algorithm, then one slot is saved by skipping the node s_7 at level 1. In such case, the next instance of the CBT algorithm would start by enabling the node with address 10 and terminate when tag τ_5 successfully sends its reply, that is, after the address 101 has been enabled. In the next section, we describe the framework that systematically estimates which tag subsets should be enabled for transmission after a single instance of CBT is finished.

8.2.4 General Arbitration Framework for Tree-Based Protocols

The key part of the framework proposed in [18] is the interpretation of the randomization procedures used in a tree protocol. Let us assume that, upon the transmission of the initial probe, each tag generates a random real number, uniformly distributed in the interval