applied sciences

MDPI

Article

A Comparison of RFID Anti-Collision Protocols for Tag Identification

Nikola Cmiljanic [1,2,*], Hugo Landaluce [1,2] and Asier Perallos [1,2]

[1] DeustoTech—Fundación Deusto, Avda. Universidades, 24, 48007 Bilbao, Spain; hlandaluce@deusto.es (H.L.); perallos@deusto.es (A.P.)

[2] Faculty of Engineering, University of Deusto, Avda. Universidades, 24, 48007 Bilbao, Spain

* Correspondence: n.cmiljanic@deusto.es; Tel.: +34-944-139-003 (ext. 3269)

Received: 30 June 2018; Accepted: 27 July 2018; Published: 1 August 2018

Abstract: Radio Frequency Identification (RFID) is a technology that uses radio frequency signals to identify objects. RFID is one of the key technologies used by the Internet of Things (IoT). This technology enables communication between the main devices used in RFID, the reader and the tags. The tags share a communication channel. Therefore, if several tags attempt to send information at the same time, the reader will be unable to distinguish these signals. This is called the tag collision problem. This results in an increased time necessary for system identification and energy consumption. To minimize tag collisions, RFID readers must use an anti-collision protocol. Different types of anti-collision protocols have been proposed in the literature in order to solve this problem. This paper provides an update including some of the most relevant anti-collision protocols.

Keywords: RFID; IoT; anti-collision; tag identification; memoryless protocols; bit tracking; window; tag ID

1. Introduction

The Internet of Things (IoT) is a set of millions of physical devices around the world that are connected to the Internet, collecting and sending data. It enables objects and machines to connect to the Internet and share collected information. Thanks to very cheap procedures and wireless networks, it is possible to turn anything, from a chip to a big building, into part of the IoT. This is an additional level of digital intelligence of devices that would otherwise be dumb, permitting communication without humans, and merging the digital and physical worlds. The term IoT was first coined by British entrepreneur Kevin Ashton in 1999, while working at Auto-ID Labs, referring specifically to a global network of objects connected by RFID [1]. Nowadays, the IoT has been the focus of numerous research studies. Moreover, although IoT extends beyond the RFID, this technology is one of the key cores to implement the IoT.

RFID technology uses radio frequency in order to identify tags, which are small sticks attached to the objects that wish to be identified [2,3]. The RFID system operates with a reader sending and receiving information from numerous tags in the interrogation range of its antenna at the same time. That is, the reader broadcasts messages using electromagnetic waves to tags in the interrogation area, while receiving responses from the tags through backscattering. Therefore, a multi-access arbitration procedure is necessary in order to prevent the response from being garbled [2,4].

When more than one tag transmits simultaneously to a reader, their backscattered signals cancel each other out, resulting in an illegible message by the reader. This is called *collision* and it causes a loss of identification time and an increase in the power consumed by the reader.

Several papers in the literature present different types of anti-collision protocols [5,6]. This paper classifies and presents an update on the existing literature collections, summarizing and qualitatively comparing some of the most novel anti-collision protocols.

The paper is organized as follows: Section 2 presents background on anti-collision protocols and the different existing types. Section 3 focuses on Aloha-based protocols, Section 4 presents some of the latest Tree-based protocols, and Section 5 presents Hybrid protocols. Section 6 contains a discussion and provides a comparison between all of the types presented. Finally, Section 7 offers a conclusion.

2. Background

This section presents an overview of the existing anti-collision protocols. To properly understand the information gathered in this work, some basic terms related to anti-collision protocols are explained. Tags' responses are encapsulated in periods of time called slots. During the identification process, and depending on the number of tag responses received by the reader, three types of slot can occur: collision, idle, and success. A collision occurs when more than one tag responds to the reader's command in the same slot. When no tag responds to the reader's command, an idle slot happens. Success occurs when only one tag is correctly read and, therefore, identified by the reader.

Communication between a single reader and several tags is based on sending and receiving information. Every communication channel has a predefined capacity, which is specified by its maximum data rate and the range of its antenna. The available channel capacity must be divided between all tags in the interrogation zone such that all of the data can be transmitted from several tags to a single reader without mutual interference. In early radio technology, the problem of multi-access arose. Many participants attempted to access a single satellite or base station. Therefore, many procedures were developed with the aim of separating the individual participants' signals from one another.

2.1. The Collision Problem

The general problem of collision in RFID can be classified into two different situations:

- Reader collision occurs when the reader attempts to make communication with tags that are in the coverage area of another reader [7]. This type of collision is shown in Figure 1. This collision causes two different problems:
  - Signal interference occurs when the fields of two or more readers overlap and interfere. This problem can be solved by programming all readers to read at fractionally different times.
  - Multiple reads of the same tag occur when the same tag is read once by every overlapping reader.
- A tag collision occurs when more than one tag attempts to transmit its ID at the same time: the reader will receive a mixture of the tags' signals and cannot understand it. This type of collision is shown in Figure 2.

Simultaneous responses from numerous tags prevent the reader from correctly translating the signal, which decreases throughput. No tag is aware of the activity of any other tag, and so they cannot prevent the simultaneous transmission of tags. The transmissions of three tags, shown in Figure 1, are not synchronized, but in many cases, the reader is synchronized with at least one tag in the interrogation zone. In the presented illustration, the reader is prevented from decoding the entire transmission and it has experienced a collision. To solve this problem, anti-collision protocols are very influential.

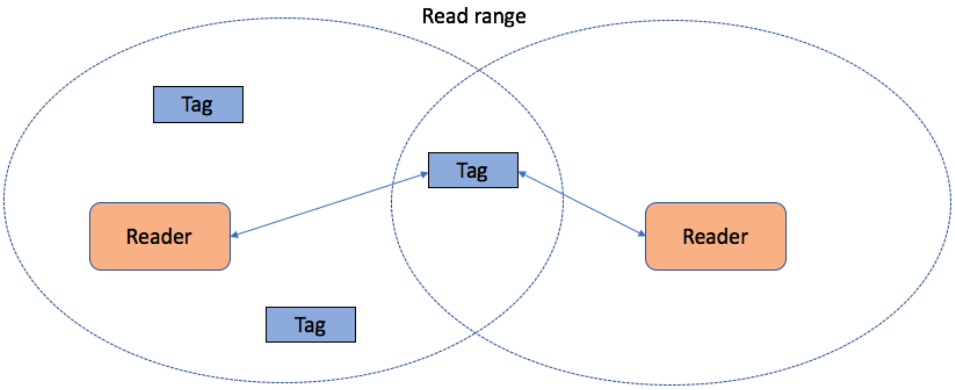

Figure 1. Reader collision.

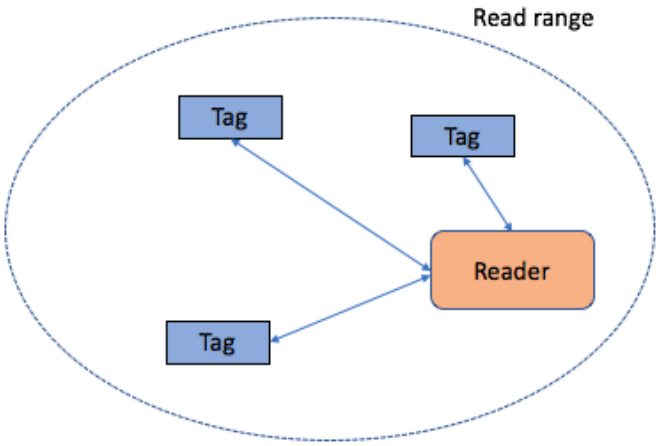

Figure 2. Tag collision.

2.2. Multi-Access Methods

Each anti-collision protocol uses certain multi-access methods for identification in order to physically separate the transmitters' signals. Accordingly, they can be categorized into four different types: Space Division Multiple Access (SDMA), Frequency Division Multiple Access (FDMA), Code Division Multiple Access (CDMA) and Time Division Multiple Access (TDMA) [2,5,8]. Figure 3 shows various multiple access and anti-collision procedures.

SDMA—The term space division multiple access relates to dividing of the channel capacity into separate areas. Protocols based on this method can point the beam at different areas in order to identify tags. The channel is spatially separated using complex directional antennas. Another means of achieving this is through the use of multiple readers. As a result, the channel capacity of adjoining readers is enhanced. A huge number of tags can be read simultaneously as a result of the spatial distribution over the entire layout. This method is quite expensive and requires complex antenna design. The use of this type of method is restricted to a few specialized applications [9–11]. This technique is shown in Figure 4.

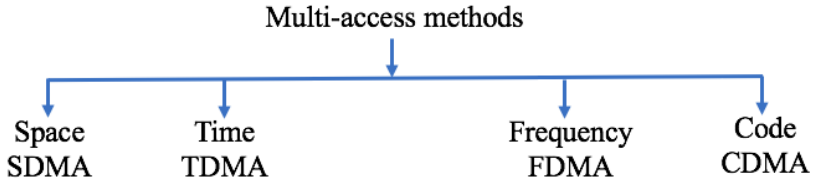

Figure 3. Multi-access methods.

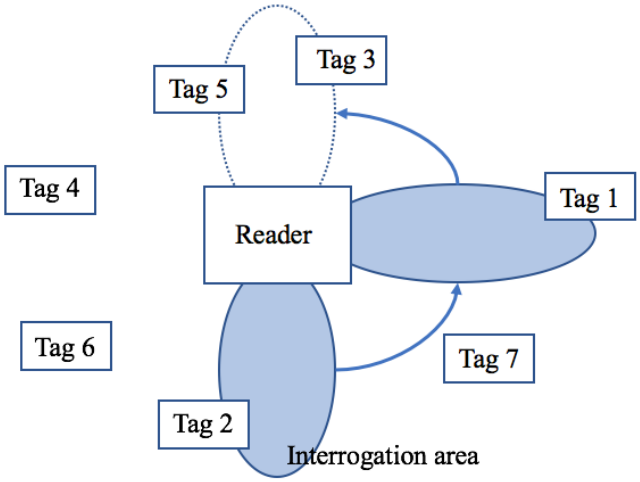

Figure 4. Space Division Multiple Access (SDMA) procedure.

FDMA—Tags transmitting in one of several different frequency channels requiring a complex receiver at the reader. Consequently, different frequency ranges can be used for communication from and to the tags: from the reader to the tags, 135 kHz, and from the tags to the reader, in the 433–435 MHz range. However, this technique is expensive and is only intended for certain specific applications [12]. Figure 5 shows FDMA procedure.

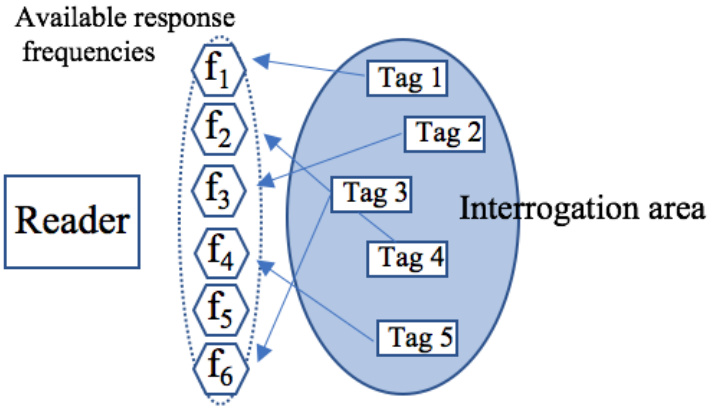

Figure 5. Frequency Division Multiple Access (FDMA) procedure.

CDMA—Requires tags to multiply their ID by a pseudo-random sequence (PN) before transmission. CDMA is quite good in many ways, such as the security of the communications between the RFID tags and the reader, and multiple tag identification. It adds great complexity and

is expensive for RFID tags. Furthermore, this method consumes a great deal of power and can be classified as a group with elevated demands [13]. Figure 6 shows this procedure.

TDMA—Given that it is less expensive, this is the most widely used method. This method involves the largest group of anti-collision algorithms. The transmission channel is divided between the participants and ensures that the reader can identify a tag at different times in order to avoid interfering with another one. The space-distributing characteristic of tags is not considered. The number of tags in the interrogation zone is reduced after every successful response. Another option involves the ability to mute all tags except for the transmitting tag. After that, the tags are activated one by one [14,15]. TDMA is shown in Figure 7.

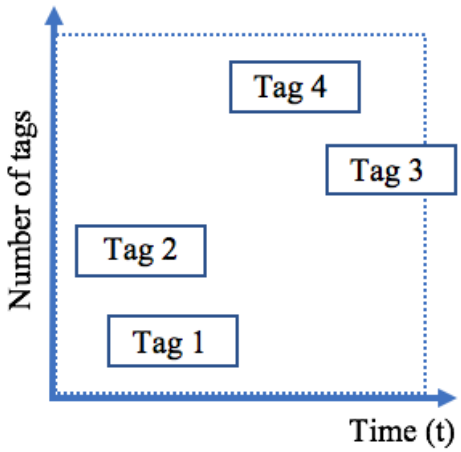

Figure 6. Code Division Multiple Access (CDMA) procedure.

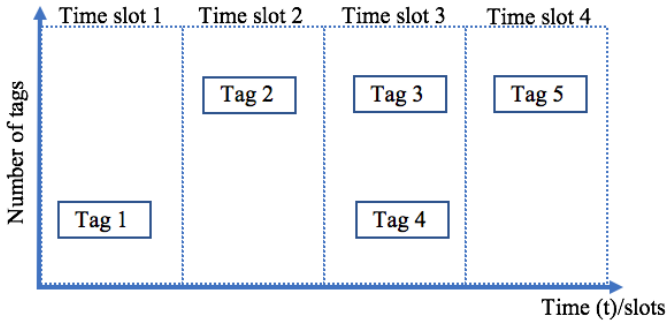

Figure 7. Time Division Multiple Access (TDMA) procedure.

In an RFID environment, anti-collision protocols typically use the TDMA method. Protocols that use this method first select an individual tag from a large group using a specific algorithm and then the communication takes place between the selected tag and the reader. Significant increases in number of collisions in the identification process decreases the throughput and increases the number of transmitted bits. These protocols can be divided into three categories: Aloha-based protocols, tree-based protocols and hybrid protocols (which use a combination of the first two methods). In the following subsections, some Aloha, tree-based and hybrid protocols shall be presented.

3. Aloha Protocols

Aloha-based protocols use a random-access strategy in order to successfully identify the number of tags in an interrogation area [16–20]. They belong to the group of probabilistic protocols because the tags transmit their own ID in randomly selected slots in a frame in order to reduce the possibility of a collision. However, there is no guarantee that all of the tags will be identified in the interrogation process. These protocols suffer from the well-known tag starvation problem, in the sense that a tag

may not be correctly read during a reading cycle due to an excessive number of collisions with that same tag. Every frame consists of a certain number of slots, and the tags can only respond once per frame [16].

The main Aloha-based protocols can be divided into four subgroups: Pure Aloha (PA), Slotted Aloha (SA), Frame Slotted Aloha (FSA) and Dynamic Frame Slotted Aloha (DFSA) protocols.

3.1. Pure Aloha

Pure Aloha (PA) is one of the simplest anti-collision protocols. It is based on TDMA [21,22]. Whenever tags enter the interrogation zone, they randomly choose a frequency on which to transmit their data. A collision will occur if several tags transmit data at the same time, resulting in complete or incomplete collisions. A complete collision occurs when the messages of two tags fully collide; an incomplete collision, however, takes place when only part of the tag message collides with another tag message. This procedure is shown in Figure 8 and will be repeated until all tags are successfully identified.

PA has been presented using different extra-features [8,23] such as: muting for silencing tags after being identified; the 'Slow down' for decreasing a tag response rate after identification; the 'Fast Mode' for sending a silence message before a tag begins transmission; and combinations of these different features.

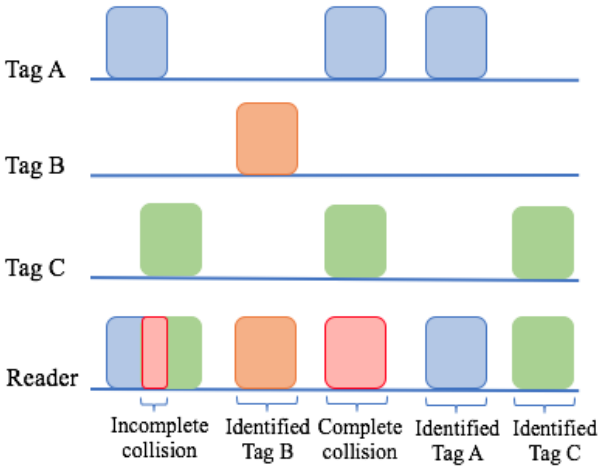

Figure 8. An example of PA.

3.2. Slotted Aloha

To avoid incomplete collisions, Slotted Aloha (SA) has been created. In SA, the time is divided into several slots and each tag must randomly select a slot in which it will transmit its data [8,21,23,24]. The communication between the reader and the tag is now synchronous. An example of communication with this protocol is presented in Figure 9.

Also, SA can use features similar to those presented for PA. The muting of slow down features are used to silence or decrease the rate of the tags; the early end closes the slot earlier than normal; and also combinations of the types.

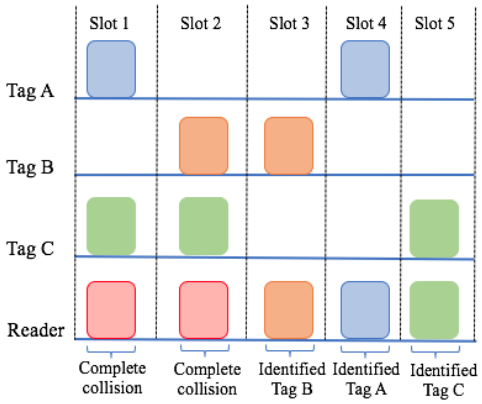

Figure 9. An example of SA.

3.3. Framed Slotted Aloha and Dynamic Framed Slotted Aloha

In Framed Slotted Aloha (FSA), the time is divided into a variable number of frames and each frame consists of several slots [16,25–28]. All tags need to transmit data into a fixed length frame, but each tag must choose only one slot in a frame to transmit data. This protocol significantly reduces the probability of collision since tags can only respond once in a frame. Before the onset of communication, the reader generates a random time that is less than the fixed frame size, to select just one slot in a frame. If a collision occurs, the involved tags will once again choose a slot in which to respond in the next frame. The main inconvenience of FSA is slot wastage when the number of tags is small and the size of the frame is significantly larger [29,30].

To ameliorate this disadvantage, the Dynamic Frame Slotted Aloha (DFSA) protocol was developed [16,31,32]. DFSA is capable of changing frame size according to an estimate of the number of tags. At the beginning of each frame, the reader informs the tags of the frame length. Every tag selects a random number $[0, F-1]$, where F denotes the frame size and all tags respond within the number of slots. At the end of the frame, the reader estimates the number of colliding tags, then adjusts F accordingly. There are certain disadvantages to tag estimation, such as: increased computational costs in the identification process and errors that degrade the protocol's efficiency. Examples of these two protocols are shown in Figure 10.

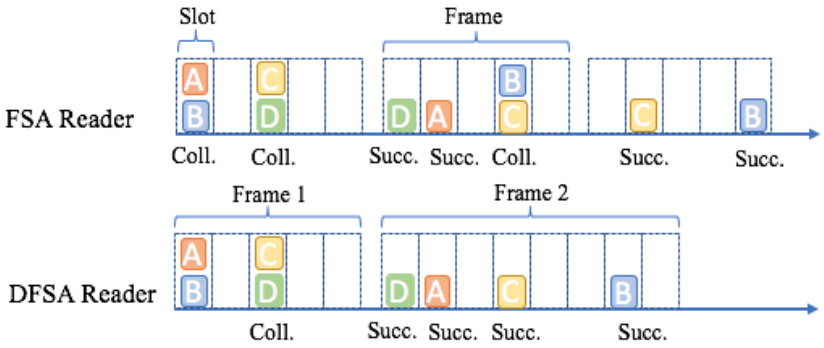

Figure 10. Example of FSA and DFSA.

Q Protocol

The Q protocol is used in the EPCglobal Generation-2 (Gen-2) standard [18,33]. The Q protocol is a DFSA-type protocol that modifies frame size using feedback from the last frame accomplished [34]. The Q algorithm can jump into the following frame without finishing the current one [34].

The Q algorithm operates with two basic parameters: Q, and a constant c that can be modified depending on the situation [24,25]. The Q. variable is an integer ranging from 0 to 15. This protocol works with three types of commands:

- Query command, transmitted by the reader to all tags in the interrogation area, in order to force all tags to choose a slot number (SN) from $[0, 2^Q - 1]$. This command initiates the identification process by providing a new value of Q.
- QueryAdjust is the command used to instruct all tags to increase, decrease, or maintain the Q value unchanged and to reselect their SN. Q_{new} denotes the last calculated Q. Accordingly, Q could be increased by c, decreased by c, or left unchanged, according to the algorithm.
- QueryRep is used in order to notify all tags to decrease their SN by 1.

The procedure of the Q protocol appears in the flow chart in Figure 11. If it is time to initiate a new inventory round, the reader will transmit a Query command. If the tags receive *Query* or *QueryAdjust*, they need to choose SN from $[0, 2^Q - 1]$. If they receive a *QueryRep*, all unidentified tags decrease their SN counter by 1. Only tags with SN = 0 will generate a 16-bit random number (RN16) and will respond with an RN16 to the reader. There are three possibilities, depending on the tags' response:

- Successful reply. If only one tag responds and the reader successfully received RN16. Subsequently, the reader sends the ACK and only the tag that successfully responded recognizes the ACK and reports its EPC to the reader.
- Collided reply. If more than one tag transmits RN16, a collision occurs. Then, the tags will increase Q_{new} by the constant c. Typical values for c are $0.1<c<0.5$. The value of c is adjusted according to the type of application. Higher values of c will provide more aggressive frame adjustments.
- No reply. When no tags respond in the slot, the reader decreases Q_{new} by c.

A similar protocol solution provides different updating steps for no-reply and collided reply by calculating the probabilities of idle slots and collided slots, without considering the parameters in Gen-2 [28,35]. Table 1 summarizes the observations regarding the Aloha protocols, where the efficiency measures the exploitation of the tags' responses and the influence of a collision on a tag response. It is calculated using the expression $\frac{n}{sl_t} \times 100$, where n is the number of tags, and sl_t denotes the total number of consumed slots.

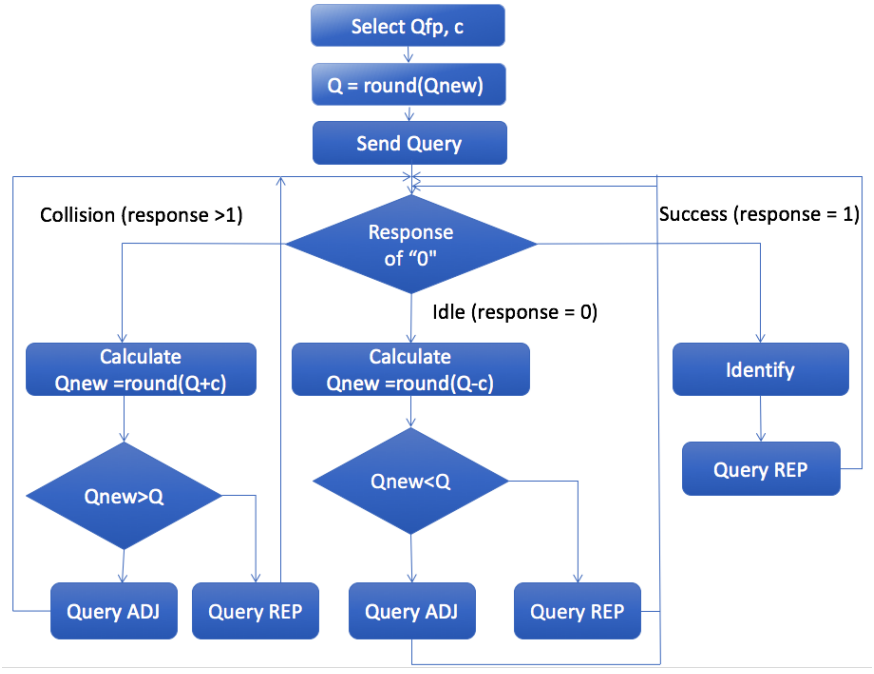

Figure 11. Flow chart of the Q protocol.

Table 1. A comparison of Aloha protocols.

Protocol Name	PA	SA	FSA	DFSA	Q
Protocol feature	Tags transmit after random time to the reader. In the case of a collision tags will retransmit after a random delay.	Tags transmit their ID in synchronous time slots. In case of a collision, tags retransmit after a random delay.	Each tag responds only once per frame.	Tags transmit once per frame. The reader uses a tag estimation function to vary the frame size.	The reader dynamically adjust critical parameter (Q) based on the type of replies from tags.
Disadvantages	In a dense tag environment the the number of collision increases significantly.	In a dense tag environment the the number of collision increases significantly. The reader requires synchronization with tags.	It uses a fixed frame size and does not change the size during the identification process.	It cannot move into the next frame at any time based on the situation of collision without finishing the current frame.	This protocol may encounter some problems (lower throughput) on adjusting Q, especially when the frame size is larger than the number of tags.
RTF/TTF	TTF	RTF	RTF	RTF	RTF
Efficicency	18.4%	36.8%	36.8%	42.6%	36.8%
System cost	Very low	Low	Expensive	Expensive	Expensive
Complexity	Very simple	Simple	Medium	High	Medium

4. Tree-Based Protocols

One of the main features of tree-based protocols is that they are deterministic, since ideally, they are designed to identify the whole set of tags in the interrogation area [36–40]. These protocols have simple design tags and work very well with a uniform set of tags.

Tree-based protocols usually work with a muting capability since they need the identified tags to remain quiet after their identification. These protocols usually work using queries, which are broadcast commands transmitted by a reader to require the tags to respond. If a tag's ID does not match the query, the reader command is rejected.

First, the most popular tree-based protocols are presented here. Then, a selected group of protocols will be presented with the common feature of the use of Manchester coding. The first group includes: Query Tree (QT), Query Window Tree (QwT), and Smart Trend Transversal (STT). Another group consists of tree-based protocols that use Manchester coding: Binary Search (BS), Collision Tree (CT), Optimal Query Tracking Tree (OQTT), and Collision Window Tree (CwT).

4.1. Query Tree Protocol

The query tree protocol (QT) is one of the most representative memoryless protocols, in which the reader must provide the tags with a query and the matching tags must respond with their full ID [41]. Tag response depends directly on the current query, ignoring the prior communication history. QT tags involve only simple hardware requirements because they only compare the reader query with their own ID and respond if it coincides. The identification process consists of more rounds in which the reader sends a query, and tags whose ID prefix match the current query respond with their whole ID binary value. In the case of a collision, the reader forms two new queries by appending q with a binary 0 or 1. New queries will be placed in a Last Input First Output stack (LIFO). If there is no response to a query, the reader knows that there is no tag with the required prefix, and the query is rejected. This kind of slot is called idle. If just one tag responds to the reader query, that tag will be identified. By extending the query prefixes until only one tag's ID matches, the algorithm can identify the rest of the tags. The identification procedure is completed when the LIFO stack is empty.

Figure 12 shows the QT protocol being used to read 6 tags (Tag A–Tag F). Each tag uses an ID length of $k = 6$ bits. Initially, the LIFO stack is empty, and the reader begins with a null string. After a collision occurs, the reader pushes queries 0 and 1 into LIFO stack. During the second round, the reader pops from the stack and transmits query 0. In the example in Figure 12, tags 000100 and 001010 match the required prefix, which causes both to transmit and collide. The reader is unable to understand the messages from the tags. The reader then pushes into the stack queries 01 and 00. In the next round, the reader transmits query 00. Again, both protocols respond with their ID and a new collision occurs. In the stack, the following new queries are added: 001 and 000. The reader transmits query 000 and only one tag responds (000100). This tag is successfully identified and will not answer any of the following reader requests. The reader then transmits query 001 in slot 4, which matches tag 001010. In the next round, the reader pops and transmits query 01. For this query, there will be no response since no tags contain that prefix. In round 7 the reader transmits query 1 and the tag from the right side of the tree responds. Four tags will receive this query and a new collision occurs. The reader experiences a collision, since tags 100011, 101110, 110110 and 111001 responded to the query 1. As a result, queries 11 and 10 are pushed onto the stack. The identification process is repeated until round 13, in which the reader transmits the last query (111) from the stack. Overall, the reader uses 13 rounds to read 6 tags.

In the literature, numerous extensions to the QT protocol have also been proposed [38,42–45].

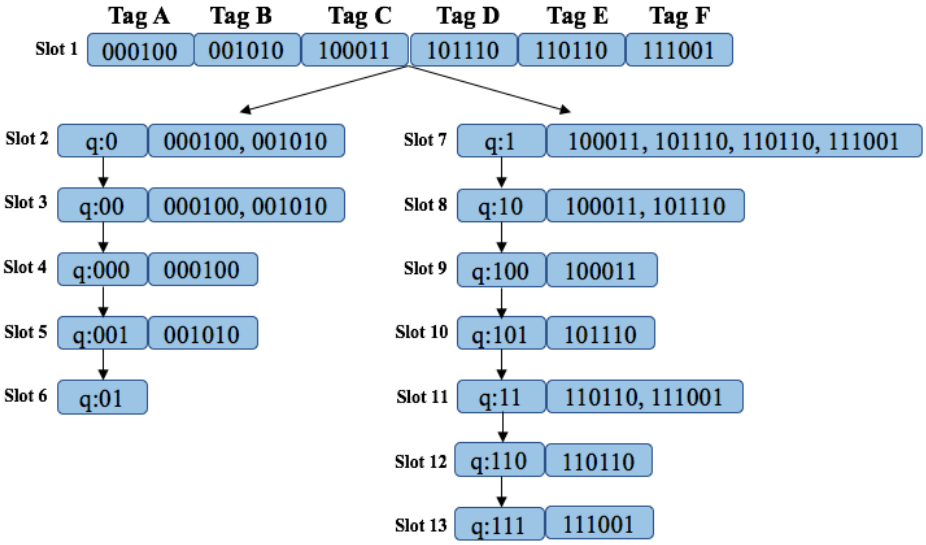

Figure 12. Example of the QT protocol.

4.2. Smart Trend Traversal Protocol

The Smart Trend Traversal protocol (STT) is a deterministic and memoryless protocol that was created with the aim of reducing the number of collisions in the QT protocol [46].

This protocol has the ability to dynamically issue queries according to an online, learned tag density and distribution. It proposes a combination of the QT protocol and the shortcutting method in order to skip a query that results in a collision. When the protocol detects the potential possibility of a collision, it will avoid it and move to the bottom level of the binary query tree. STT provides trend recognition.

The reader keeps track of the tag density and distribution in order to issue subsequent queries, and consequently, minimizes the number of empty slots and collision slots.

In this protocol, it is not necessary to have any prior knowledge of the network, and it outperforms the existing protocols. The ideal number of queries can be the total number of single nodes. The ideal queries group, referred to as the query traversal path (QTP), is denoted by $Q = q_1, q_2, q_3, \ldots, q_n$, where q_n is the last query used in the identification process [47].

It is difficult to achieve, but is desirable to get close to its value. The reader can calculate the subsequent queries depending on the tag response, which can be classified into three types:

- A collision occurs when the QTP is at too high a certain level and should be moved down by adding a longer prefix to the query. Consequently, the reader appends t bits of 0's to the last query, where $t = s + n_{col} - 1$. Let s denote the minimum increase, and n_{col} be the number of consecutive colliding slots.
- An idle slot occurs when no tag responds to a reader query. QTP needs to traverse up just one level, which can lead to a new collision. This rule will be applied only to the right side. If the empty response comes from the left side of the tree, QTP must move horizontally to the right. The reader will decrease the query length by m bits, where $m = s + n_{emp} - 1$ and n_{emp} is the number of consecutive idle slots.
- Upon a successful response, a single node is visited, indicating that the tag has been successfully identified by the reader. Then QTP moves to the symmetric node if the query finishes with a 0, but it returns one level if the query finishes with a 1.

The identification process of the STT protocol, which was explained above, is depicted in Figure 13 with 4 tags.

In conclusion, STT significantly reduces the number of collisions, the identification time, and the energy consumption as compared to the existing Aloha-based and tree-based protocols.

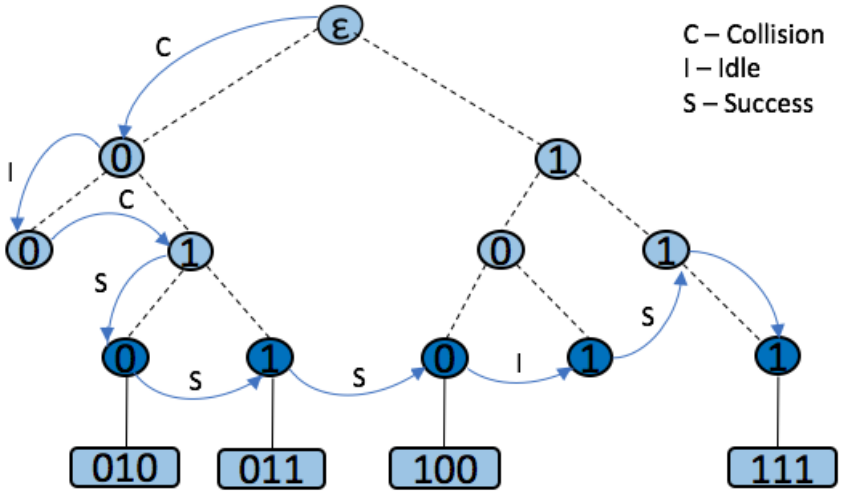

Figure 13. An example of STT protocol.

4.3. Window Based Protocols

In the majority of tree-based protocols, tags respond with their full ID or with the bits from the last query, when the query sent by the reader matches the tag ID prefix. Figure 14 shows an example of a communication slot between the reader and the tag. To reduce the number of bits transmitted by the tag, a window method has been proposed [48–50]. In the identification process, many slots ultimately collide, resulting in a huge waste of bits. Protocols using the window method reduce the number of bits transmitted by the tags. The window is defined as a bit-string of length ws bits transmitted by a tag in a slot. This bit-string is computed on the reader side, respecting the condition $0 < ws < k$. It is shown in Figure 15. Most tree-based protocols use a fixed tag response during the identification process, but some use different operational process methods with a dynamic response that is based on window synchronization.

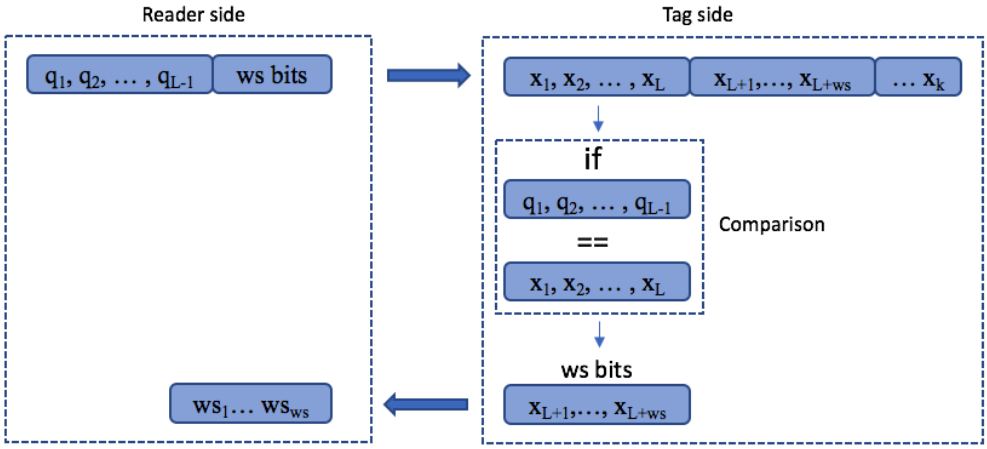

Figure 14. Example of a communication slot between the reader and one tag.

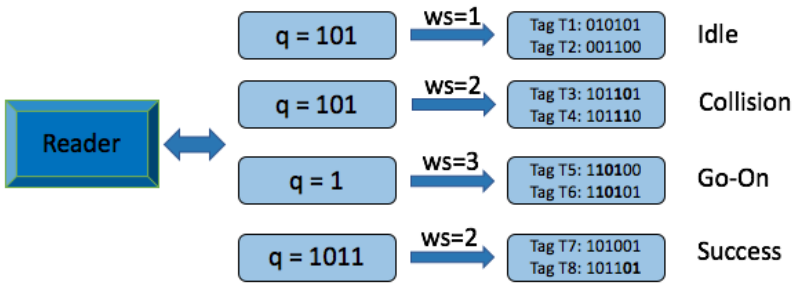

Figure 15. Window synchronized answer.

Query Window Tree Protocol

The Query window Tree protocol (QwT) is a memoryless tree-based protocol that applies a dynamic bit window to QT [48,50]. Tags respond directly depending on the current query. QwT tags compare their ID value with the query received and transmit a certain number of bits, managed by the reader. This reduces the complexity of passive tags, their energy consumed, and the identification time. A reader and tag flow chart for QwT are shown in Figure 16a,b. When tags appear in the interrogation area, the reader will broadcast to them by transmitting a query length of L bits. Tags will respond if their ID prefix matches the query sent by the reader, but with the previously specified number of bits. One of the main features is that the total number of collisions is decreased by transforming potential collisions into partial successful slots. This is a new type of slot, called go-on slots. The previously explained window methodology is implemented in the QwT protocol. The window allows tags to transmit only the bit-string instead of their full ID. If tags match a reader query, they will synchronously transmit the next adjacent ws bits of the ID. This protocol uses cyclic redundancy check (CRC) in order to differentiate between the types of tag responses. Accordingly, the slot types that can occur in the QwT protocol can be classified into 4 groups:

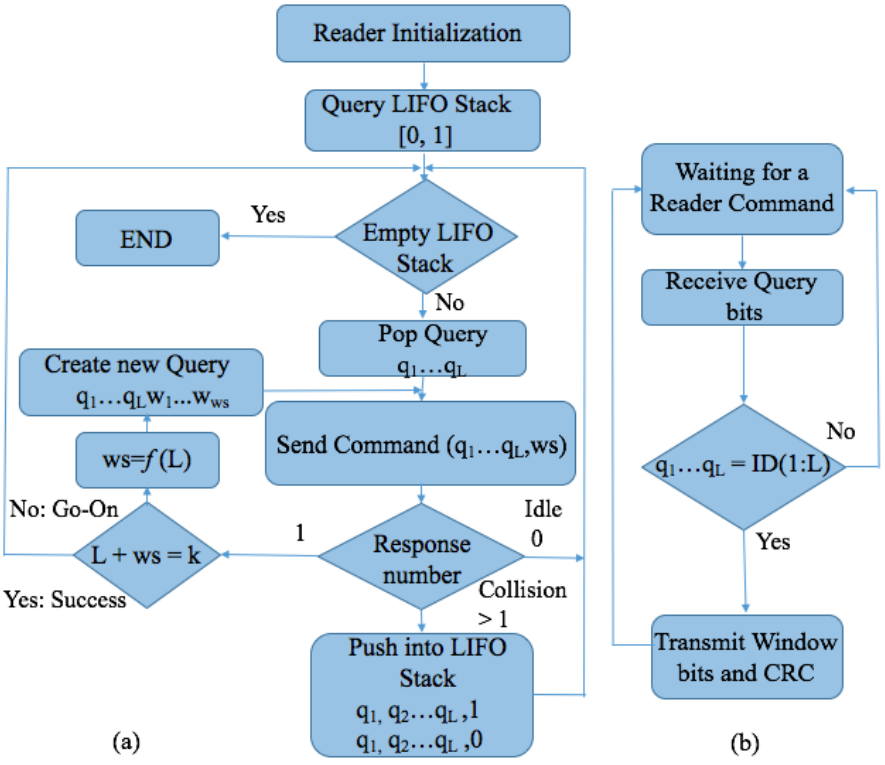

Figure 16. Flow chart of QwT protocol: (**a**) for reader; (**b**) for tags.

- Collision slot. When the reader cannot differentiate the answer, they will create two new queries by appending '0' and '1' to the former query [q_1,q_2...q_L]. The window size ws, will remain unchanged, with the value used in the previous query.
- Idle slot. When there is no response, the reader will discard the query and retain the same ws as that of the last command.
- Go-on slot. This occurs when at least one tag responds with a window and the reader is able to understand it. If $L + ws < k$ is not true, the reader will transmit a new query created from the former query and received window. During this query, the reader will append an updated ws value.
- Success slot. This is a type of go-on slot where the reader successfully receives the last part of the tag ID and $L + ws = k$. Then, the reader can save the tag, calculate the new ws, and continue with the identification process.

Using the QwT protocol, the reader computes ws using the expression (1), where β is an adjustable parameter. This heuristic function is used to provide dynamism to the value of ws. It can only be applied to the go-on and Success slots, since in a Collision or Idle slot, ws will be held unchanged. The proposed protocol maintains the memoryless feature of QT in that it is an applied bit window procedure. It provides a decrease in the number of tag-transmitted bits, but increases the number of slots and reader-transmitted bits. Altogether, this tree-based protocol achieves significant energy savings and a reduction in identification time.

$$f(L) = k(1 - e^{-\beta L}), \quad 0 < L \leq k \tag{1}$$

- A modification of the QwT is presented in [51] called Standardized Query window Tree protocol (SQwT). This protocol aims to reduce the number of bits that define ws by using a standardized value of 3 bits and approximating it to the nearest power of 2, using $s = log_2 ws$ instead. Tags calculate the number of bits to respond by using $ws = 2^s$. By using only 3 bits, SQwT can cover window size values from 1 to 128 bits.
- Another modification of QwT is presented in [52] and is called Flexible Query window Tree protocol (FQwT). This modification takes advantage of the window to perform an estimation of the tag ID distribution in the interrogation area and improve the identification time of the protocol. The functioning of the protocol is divided into two phases: the estimation of the distribution, and the identification process. During the former phase, the reader estimates the tag ID distribution until the first tag is identified (see Figure 17).

After the first tag is identified, the reader begins the identification phase, similar to that of the SQwT. This phase, however uses a different heuristic function when a go-on slot occurs, taking advantage of the c_g parameter (see Equation (2) related to the type of tag ID distribution).

$$f(c_g, L) = \frac{c_g}{L} k(1 - e^{-\beta L}), \quad 0 < L \leq k \tag{2}$$

Equation (2) is adjusted with a value of parameter β, preselected to decrease the energy consumed by the proposed protocol. Table 2 compares some of the standard tree-based protocols.

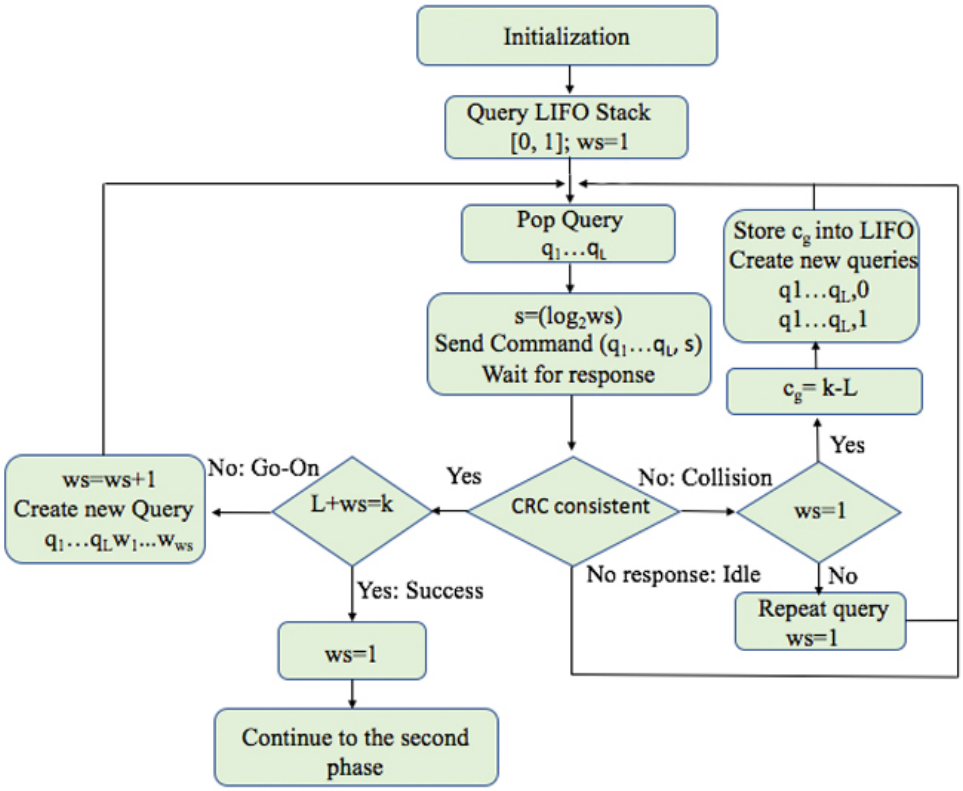

Figure 17. Flow chart of the tag ID estimation phase of the proposed FQwT protocol: (**a**) for reader; (**b**) for tags.

Table 2. A comparison of standard tree-based protocols.

Protocol Name	QT	STT	QwT
Protocol feature	Tags respond when their ID match the query transmitted by the reader.	Defines the generation of the new queries according the several predefined rules. These rules take into account the number of consecutive collisions and slots.	The reader transmits the number of bits tags must respond in addition to the query. This number is calculated using an heuristic function at the reader side.
Disadvantages	It produces a high number of collisions, particularly in the beginning of the identification procedure.	It transmits the full tag ID with each response, therefore a high number of bits are wasted with every collision.	The reader command needs a high number of bits to represent the size of the tags' responses.
Efficiency	34.6%	34%	35%
System cost	Very low	Expensive	Medium
Complexity	Low	High	Medium

4.4. Manchester Coding

Some tree-based protocols work with Manchester coding, which can be used to locate bits that have collided [2,5,53]. The use of Manchester coding to trace the collision to an individual bit is called bit-tracking in the literature [54]. In Manchester coding, the value of a bit is defined by the change in the voltage level: a negative or positive transition. A logical 0 is coded by a positive transition; a logical 1 is coded by a negative transition. In the case in which a minimum of two tags simultaneously

transmit bits with different values ('0' and '1'), the positive and negative transitions of the received bits violate the coding rules, and a collision can be tracked.

As shown in Figure 18, Tag 1 is 1100 and Tag 2 is 1010. Both tags synchronously transmit data. The reader can understand the first bit, but the second and the third bit cause a collision. The reader detects a violation of the Manchester codification on those bits, and this is interpreted as a collision located at bits 2 and 3.

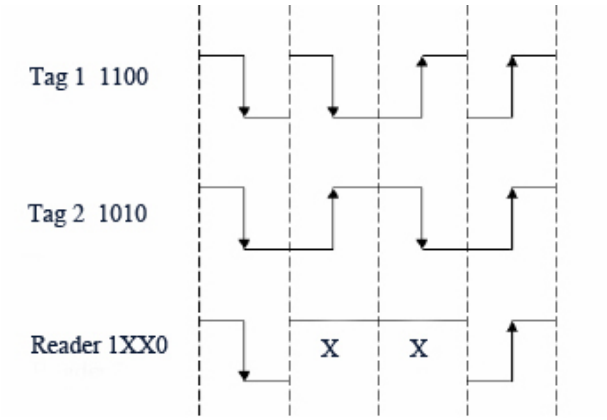

Figure 18. Example of Manchester codification showing a collision at the 2nd and the 3rd bits.

4.4.1. Binary Search Protocol

The procedure in the Binary Search protocol (BS) algorithm [2,55] involves transmitting a serial number from the reader to all the tags in the interrogation area. Only tags which have an equal or lower ID value than the received serial number will respond to the request. Then the reader checks the tags' responses bit by bit using Manchester coding and if a collision is detected, the reader divides the tags into subsets based on the collided bits.

Table 3 shows an example of BS being used to read four tags (Tag A to Tag D). The reader begins by interrogating tags with the maximum ID value 111. Tags with a value of less than 111 will respond to the query. Their answer results in collision XXX, where all three bits have experienced a collision. In the next slot, the reader transmits a new query by replacing the most significant collided bit (MSB) with a 0. The reader transmits a new query, 011, in the next slot, and all tags compare their ID with the received value. Communication in this slot again results in a collision (01X). In the second slot, the reader replaces the third bit of the command with a 0 and transmits the next query, 010. In the new interrogation round (slot 3) only Tag A has a value equal to or lower than 010, and therefore it is successfully identified. After this slot, the reader restarts the query value with the initial value 111 and transmits it. This procedure is repeated until all of the tags are identified.

Table 3. Example of the BS protocol.

Slot Number	Reader Command	Tag A (010)	Tag B (011)	Tag C (100)	Tag D (110)	Result	Type of Slot
Slot 1	111	010	011	100	110	XXX	Collision
Slot 2	011	010	011			01X	Collision
Slot 3	010	010				010	Success
Slot 4	111		011	100	110	XXX	Collision
Slot 5	011		011			011	Success
Slot 6	111			100	110	1X0	Collision
Slot 7	101			100		100	Success
Slot 8	111				110	110	Success

This protocol has two additional versions: Enhanced BS protocol (EBSA) and Dynamic BS protocol (DBSA) [56]. The main difference from EBSA is that it does not restart the reading procedure after a tag is identified, as in the basic version of BS. To reduce bit consumption, in the initial slot, the reader transmits only '1' instead of all '1's. In the DBSA version, the reader uses the knowledge from the last slot and reduces the number of transmitting bits. For example, if the reader has received 01X, it will request the tags to transmit only the last bit, since the initial prefix has already been identified.

4.4.2. Collision Tree Protocol

The Collision Tree protocol (CT) is an improvement of QT which uses bit-tracking technology in order to find which bits have collided as well as where they are [54,57]. The reader, using the bit-tracking technology, can trace a collision to an individual bit and get the correct bits successfully. This feature works using Manchester coding, which can locate the conflicting bits based on voltage transitions.

The basic features of this protocol is that it decreases collision slots and eliminates idle slots. This contributes to improved results in terms of latency and the number of bits transmitted.The advantage of this protocol compared to the QT protocol is that CT has no idle slots and reduces collision slots.

Figure 19 reveals how this protocol works in an environment with 4 tags. At the beginning of the identification process, the reader generates two queries '1' and '0' into a LIFO stack. Then, the reader pops query '0' from the stack and transmits it to the tags. In this case, one tag (010110) matches the query and responds with its ID, and the tag is identified. Then, the reader sends a new query from stack '1' and a collision occurs. Through bit-tracking, the reader can find the colliding bits and thereby resolve potential collisions. The reader pushes two new queries '11' and '10', and firstly transmits '10'. The second tag is identified (101010). On the next transmission, a collision once again occurs. The reader can trace the collision to the fourth bit. Two new queries are made: '1111' and '1110'. These are the last queries in the interrogation round because both tags (111011, 111101) are identified. From this example it may be noted that there are no idle slots and the number of collision slots and latency are reduced, which is the basic aim of the CT protocol.

In conclusion, CT is a stable and efficient anti-collision protocol for RFID tag identification. The performance of CT is very dependent on the total number of tags in the interrogation area.

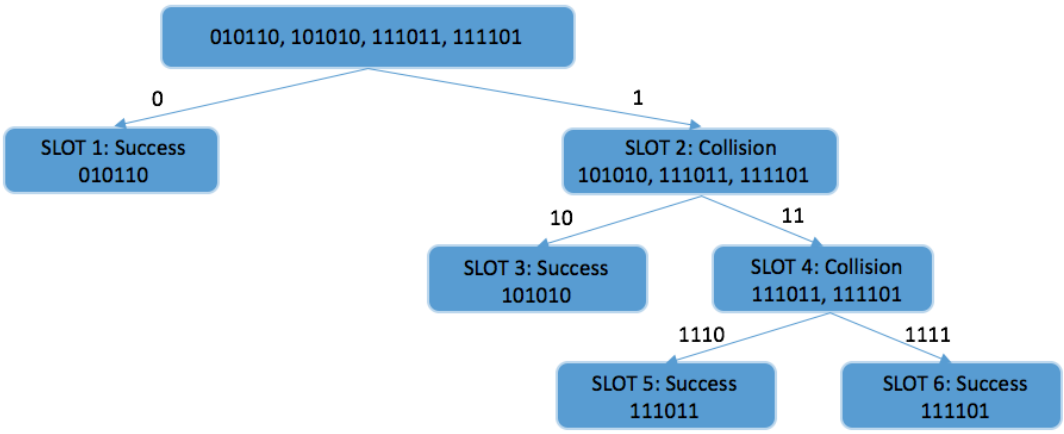

Figure 19. Example of the CT protocol.

4.4.3. Optimal Query Tracking Tree Protocol

The Optimal Query Tracking Tree protocol (OQTT) divides all of the tags in the interrogation area into small tag sets in order to reduce the number of collisions at the beginning of the identification

process [58]. This protocol uses three main approaches: bit estimation, an optimal partition, and a query tracking tree.

With bit estimation, the reader, using bit tracking technology, can estimate the number of tags in the interrogation area with a small deviation. This phase detects the status of the bits to perform the estimation. The reader broadcasts the parameter l, which denotes the default value of the tag ID length. After receiving a command, all tags must randomly choose a value k between 0 and $l-1$. To simplify the procedure, the tags only respond with a bit string of length b, instead of a bit string of length l. All tags generate a b-string of all "0" and set the bit k mod b to "1". Accordingly, the reader can compute the number of selected bits (NSB), and the number of non-selected bits (NNB) from the tags' responses. The probabilities of bits being selected or of being non-selected are calculated from the expressions presented in (2). Finally, the optimal estimation of the number of tags is denoted by $\tilde{n}$ and is calculated from (3).

$$\begin{aligned} Pr_{NB}(n,l) &= (1-\frac{1}{l})^n \\ Pr_{SB}(n,l) &= 1 - Pr_{NB}(n,l) \end{aligned} \tag{3}$$

$$\tilde{n}(l, N_{NB}, N_{SB}) = \arg\min_n \left| \begin{pmatrix} l \times Pr_{NB}(n,l) \\ l \times Pr_{SB}(n,l) \end{pmatrix} - \begin{pmatrix} N_{NB} \\ N_{SB} \end{pmatrix} \right| \tag{4}$$

The next approach is an optimal partition which determines the number of initial sets. The reader divides the tags into different sets with initial queries.

The query tracking tree is the last procedure in OQTT, and splits the set of collided tags into two subsets using the first collided bit of the tags' responses. This procedure is followed until no more queries remain in the stack.

The queries in the optimal partition are calculated using the equations $c_1 + c_2 = m$ and $2c_1 + c_2 = 2l$, where c_1 denotes the number of $(l-1)$-bit queries and c_2 the number of l-bit queries. The number of bits in each query is computed from the equations $2l - 1 < m \leq 2l$.

Table 4 shows an example of OQTT. In the interrogation area there are 5 tags, whose IDs are as follows: '1010', '1100', '0100', '0010' and '0111'. At the beginning of the frame, the reader estimates the number of tags, which here is $\tilde{n} = 5$. By using this estimate for the number of tags, the initial number of sets is calculated with the formula $m = \lceil 0.595824 \times \tilde{n} \rceil$, which yields 3. When $m = 3$, there are two 2-bit queries and one 1-bit query. The reader generates the queries 00, 01 and 1 and pushes them into the stack. In the presented example, $c_2 = 2$ (queries 00 and 01) and $c_1 = 1$ (query 1). As presented in Figure 11, when the reader pops a query, the tags answer with the value $k - q$, where k and q are the lengths of the ID and the query, respectively. When a collision occurs, this protocol splits the collided query according to the first collided bit. This is the case with slots 3, 4, 5, and 6.

Table 4. Example of the OQTT protocol.

Slot	**Query**	**Tag Response**						
		Tag A (1010)	**Tag B (1100)**	**Tag C (0100)**	**Tag D (0010)**	**Tag E (0111)**	**Status**	**Stack**
	Est.							**00,01,1**
1	00				10		Identified	01,1
2	01			00		11	Collision	010,011,1
3	010			0			Identified	011,1
4	011					1	Identified	1
5	1	010	100				Collision	10,11
6	10	10					Identified	11
7	11		00				Identified	empty

However, OQTT may incorrectly estimate the tag number. The estimation error however is negligible, only producing an imperceptible difference in the final result of the number of queries. According to the literature, e.g., [58], this protocol provides an efficiency of approximately 0.614 and is one of the most efficient anti-collision protocols for tag identification. Although the slot efficiency obtained by OQTT is very high, the preprocessing increases the energy consumption of the protocol, especially in dense tag environments [58].

- A modification of the OQTT is presented in [59] called Optimal Binary Tracking Tree (OBTT). This modification implements the estimator of the OQTT with a simple Binary Tree (BT) protocol [60]. The estimator establishes the inital upper bound for tags' counters. This produces a separation of the existing tags in groups avoiding excessive responses in the beginning of the interrogation procedure.

4.4.4. Collision Window Tree Protocol

The Collision window Tree protocol (CwT) is the second proposed window-based protocol that applies a dynamic window to CT [49,50]. This protocol adopts two techniques: bit tracking and bit windowing. The bit tracking uses Manchester coding in order to identify the colliding bit in the tags' responses. This technique avoids using the CRC, which QwT used in order to identify the type of slot. This protocol does not remove idle slots as CT does, but instead decreases the total amount of bits transmitted by all the tags.

The reader interrogates tags by transmitting a query $[q_1...q_L]$ of length L, attached to the ws of length $[\log_2 ws] + 1$ bits. The bit-string ws informs the tags of the number of bits that they must send in their reply. The variable ws is computed in every slot and is transmitted together with the query. Only matching tags transmit ws to the last query bit received, $[t_L + 1...t_{ws} + L]$ of their ID $[b_1, b_2...b_k]$. When the reader transmits a query, three possible slot statuses can take place after a tag's response:

- A collision slot occurs when at least one colliding bit is found. Then the reader creates two additional queries $[q_1, q_2...q_L, w_1...w_{col}-1, 0]$ and $[q_1, q_2...q_L, w_1...w_{col}-1, 1]$, using bit tracking. The supplementary queries are made from the last transmitted query $[q_1...q_L]$ appended with the received bits $[q_L, w_1...w_{col}-1]$ indicating the first colliding bit.
- A go-on slot occurs when at least one tag responds and the expression $L + ws < k$ is met. Then, the reader creates a new query based on the former one and the received window from the last slot. The next ws is calculated using the heuristic function in Equation (1).
- A success slot occurs when the reader checks the expression $L + ws = k$ and if it matches, the tag is successfully identified.

A CwT flow chart is presented in Figure 20. The reader transmits the first query (0) with $ws = 1$. The matching tags respond with ws bits and the reader looks for a colliding bit. After slot identification, the reader creates a new query and calculates the value of ws, depending on the type of slot.

The CwT provides a significant decrease in the number of tag-transmitted bits, but this benefit comes with a certain decrease in the number of slots and the reader-transmitted bits. This protocol achieves important energy savings due to the reduction in the time required by the tag transmission process [50].

Table 5 compares some of the tree-based protocols.

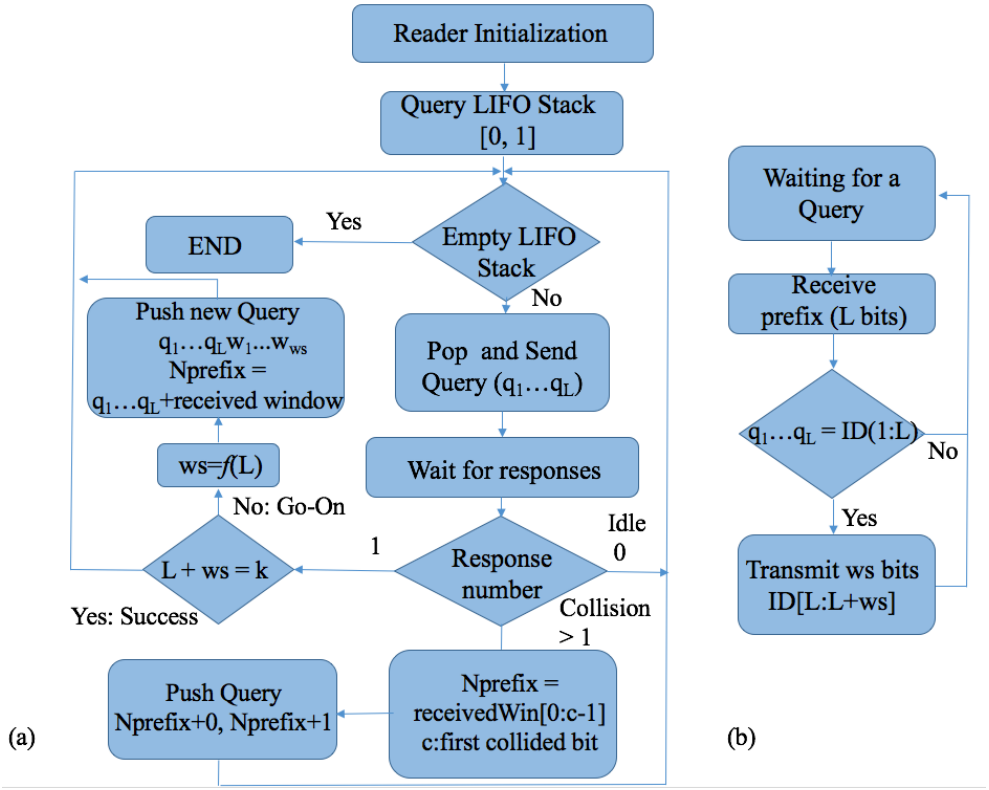

Figure 20. Flow chart of CwT protocol: (**a**) for reader; (**b**) for tags.

Table 5. A comparison of tree-based protocols using bit-tracking.

Protocol Name	OQTT	BS	CT	CwT
Protocol feature	The reader performs an initial estimation to split the tags in groups. These groups are identified using the CT protocol.	The reader transmits a serial number. Only tags having an equal or lower ID value than the received serial number will respond on request.	It is an improvement of QT which uses Bit tracking technology in order to find which bits collided as well as where they are.	The reader is able to identify the collided bits from the tags' responses, and uses them to calculate the new tag response size.
Disadvantages	Very complex protocol. Hard to be physically implemented.	The reader restarts the reading process after a tag is identified.	It wastes a high number of tag bits on every collision, increasing the energy consumed by the reader during the process.	The reader command needs a high number of bits to represent the size of the tags' responses.
Efficiency	61.4%	30%	50%	35%
System cost	Very expensive	Low	Medium	Medium
Complexity	Very high	Low	Medium	Medium

5. Hybrid Protocols

Hybrid protocols combine the advantages of tree-based and Aloha-based protocols to avoid their problems and provide better features in tag identification [61–64]. Most of them first implement a tree-based procedure and tag estimation procedure in order to predict the number of tags. Therefore, the combined Aloha-based and tree-based protocol procedures are known for their high

complexity and hardware demands. This kind of protocol can significantly increase performance as compared to the previous ones.

Recent proposals include the Tree Slotted Aloha (TSA) and Binary Tree Slotted Aloha (BTSA). TSA uses a tree structure, and the tag's responses are organized in slots, as in FSA. In the BTSA protocol, tags randomly choose a slot after the reader query.

5.1. Tree Slotted Aloha

Tree Slotted Aloha (TSA) is a probabilistic protocol created to reduce the number of collisions occurring in FSA [62]. When more tags collide in a slot, FSA attempts to solve this problem in the next frame. However, in the new approach, if more tags collide in a frame, only those tags that are involved in that collision are queried in the next slot.

TSA uses l_o—estimation for the initial frame size. This protocol provides very good efficiency, despite the fact that this number can be far from the actual number of tags.

The initial query consists of a request for data by specifying the frame size l_i. Then, all tags in the interrogation area will generate a random number in the range [0, l_i] and transmit its ID in that randomly selected slot.

The protocol is organized in a tree structure. The first node in a tree is the first interrogation round. The reader sets the initial frame with the following data: l_0, N_i represents the number of transmitting tags in slot i, where $i \leq l_0$, $N_i \geq 0$, and $\sum_i N_i \geq n$ must hold. If$N_i \geq 2$, there is a collision in slot i.

At the end of each interrogation round, if the reader detects a collision, it begins a new frame from each slot where the collision was detected. This is accomplished by adding new nodes to the tree: every new node is a son-frame of the collided slot. In each round, the tags store the generated random number from the previous round and increase their tree level counter by 1 so they will know when they should transmit. Every time the reader detects a collision, it creates a new node in the tree and a new round involving only the tags that have collided in that slot. This procedure is shown for the example in Figure 21.

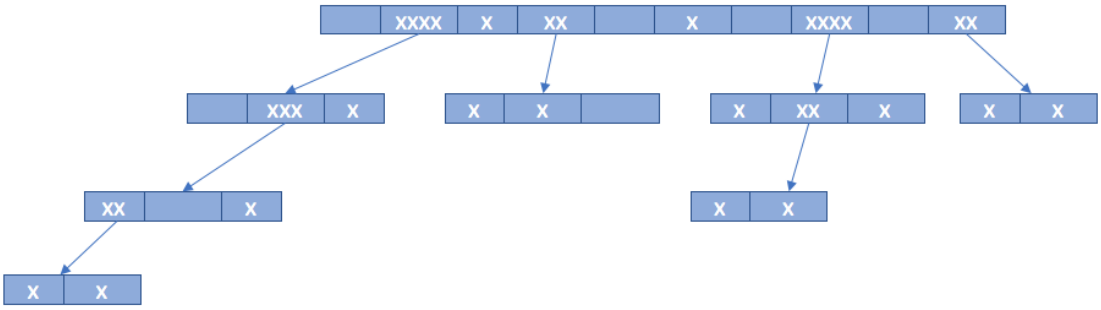

Figure 21. An example of Tree Slotted Aloha (TSA).

TSA is a modified version of the FSA protocol, created to reduce the number of collisions. This protocol behaves better than FSA. TSA achieves an efficiency of between 37% and 41% [62].

5.2. Binary Tree Slotted Aloha

The reader in the Binary Tree Slotted Aloha (BTSA) uses a dynamic frame length adjustment and BTSA algorithm [65]. Each tag from the interrogation area randomly chooses a slot and transmit its ID. If the reader successfully identifies a tag it will not be activated in the subsequent slots. When a collision occurs, the collided tags are resolved by binary tree splitting, while the rest of the tags will wait until that process is successfully completed.

The collided tags are continually split into two sets until each set has only one tag. This operation is performed by the BT. The initial frame length is $L = 2^Q$ and the highest efficiency is achieved when the initial frame size is close to the number of tags. Since BTSA has no estimation of the tag set

size, the reader cannot set the initial frame size according to the number of tags. Some protocols are presented in order to achieve higher efficiency in a wide range of number of tags. An example of BTSA is shown in Figure 22.

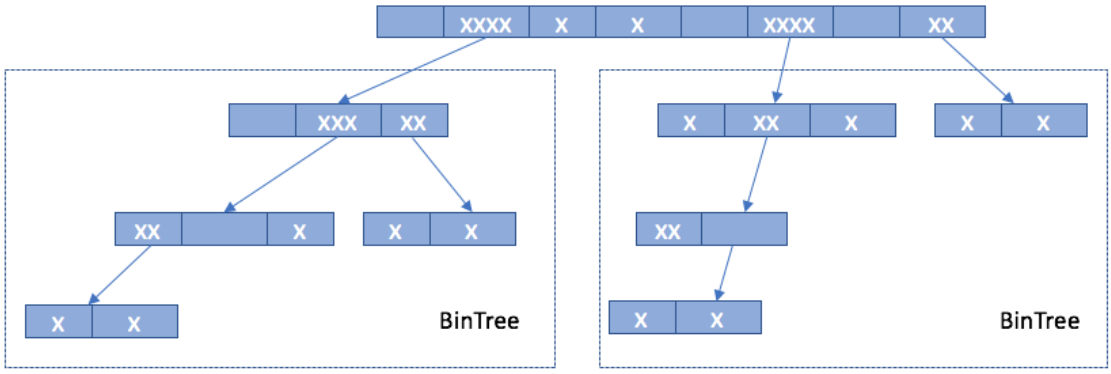

Figure 22. An example of Binary Tree Slotted Aloha (BTSA).

5.2.1. Dynamic Binary Tree Slotted Aloha

Dynamic Binary Tree Slotted Aloha (dynamic BTSA) involves a dynamic frame adjustment and the basic BTSA algorithm [65]. The advantage of this protocol is that the reader can adjust its frame size by judging only the first slot type in the identification process.

Figure 23 shows the dynamic BTSA algorithm, where the initial frame length is $L = 2^Q$, and the initial $Q_0 = 4.0$. The procedure is very similar to the Q protocol. Firstly, the reader transmits a QueryAdjust command with frame length to all of the tags in the interrogation area. Subsequently, each tag randomly chooses a random number between 0 and $L-1$. Tags whose Counter value is 0 transmit their ID. Then the reader transmits a new request with a new L and will be capable of receiving responses in the first slot of the following frame. If the first slot is idle, the reader decreases the value of Q by 1 ($Q = Q-1$) and the reader creates a new frame based on the updated Q. If the reader successfully identifies a tag in the first slot, Q will not be changed and the reader will move to the BTSA algorithm [65].

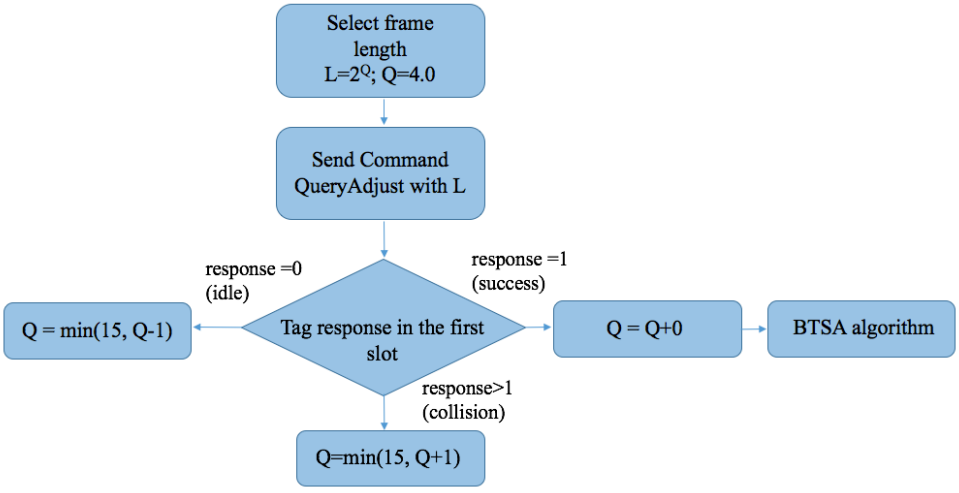

Figure 23. Flow chart of Dynamic BTSA.

In BTSA, the reader transmits the Query command in a frame. The reader has a slot counter (SC) that is set to 0 at the beginning of the frame, and is increased by 1 at the end of each slot. When the

frame length is equal to SC, the frame finishes. When the reader receives an ID in a slot, it knows the type of slot and will inform other tags by transmitting its feedback. If the reader detects a collision in a slot, it will resolve the collided tags by BinTree splitting [66]. In order for the reader to know when the binary tree has finished, it uses the variable B The initial value of B is set to 2. In the case of a collision, B is increased by 1, and if there is no collision, B is decreased by 1. Only if $B = 0$ does the reader know that the binary tree has finished.

Dynamic BTSA reduces the number of collisions and improves identification efficiency [65].

5.2.2. Adaptive Binary Tree Slotted Aloha

Adaptive Binary Tree Slotted Aloha (Adaptive BTSA) offers an improvement to the Q protocol [65]. This protocol adjusts the frame size based on the tags' responses in a current slot. Adaptive BTSA first uses features from the Q protocol. If there are numerous collisions in a frame, the reader ends the frame earlier and transmits a new command with a new frame length. If there are excessive idle slots, the reader once again ends the frame earlier and sends a new command with a smaller frame length.

The reader uses the parameters B and Q_{fp} in order to calculate the frame length. The initial frame length is to $L = 2^Q$ and $Q = 4$. The Q algorithm can adjust the frame length by adjusting Q. The value of Q. is the rounded value of Q_{fp}, which is a floating representation of Q. In the following process, the reader dynamically adjusts each slot using the presented values c [34]. In the first slot, if a collision occurs, the reader will calculate Q_{fp} by increasing it by c. In the case of an idle slot, the reader decreases Q_{fp} by c. When the reader identifies a tag, it will not change Q_{fp}. The flow chart of this protocol is shown in Figure 24.

The function framesize(Q) shown in Figure 24 denotes that a new frame has started and its length is 2^Q. Adaptive BTSA combines the Q algorithm and the BinTree strategy. The main difference between Adaptive BTSA and the Q protocol is that when a collision occurs in the slot, the collided tags will be resolved by BinTree.

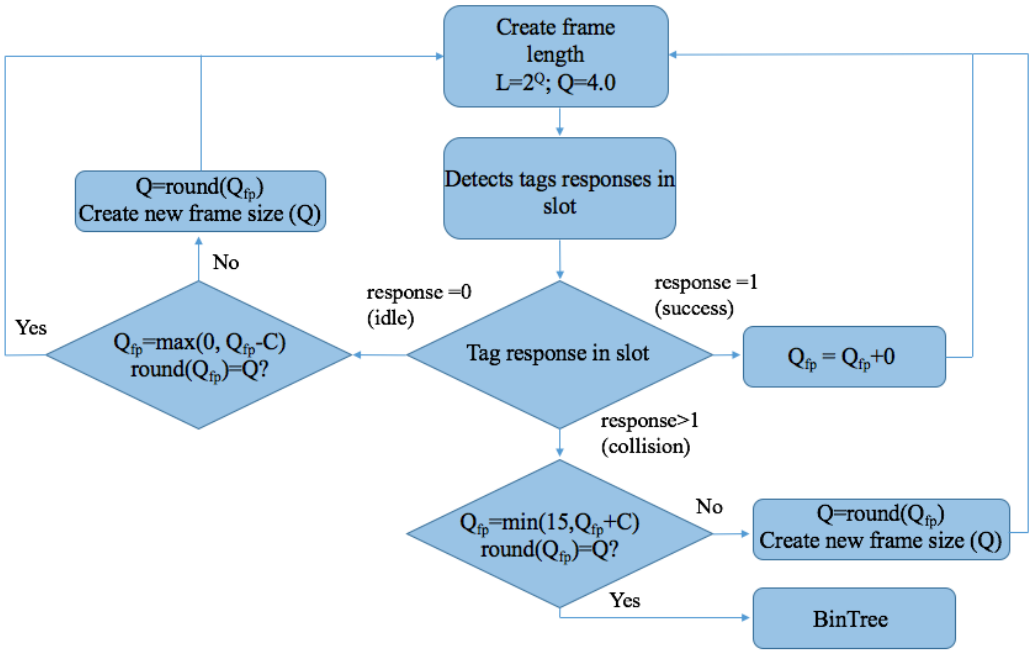

Figure 24. Flow chart of Adaptive BTSA.

6. Discussion

Anti-collision protocols are a critical part of any RFID system. This section offers a critical analysis of the different protocols presented in the previous sections. The tag collision problem results in the

wasting of bandwidth and energy, and an increased latency. Thus, an optimized anti-collision protocol is essential for a competitive RFID system.

The breadth of the literature reveals that there has been a great amount of research conducted in this area. There are two main types of anti-collision protocols: deterministic and probabilistic. In probabilistic protocols, the tags transmit their own ID in randomly selected slots in a frame in order to reduce the possibility of a collision. The tag answers are distributed into the slots and all of them have a chance of being identified. These type of protocols are highly adaptable to the appearances and disappearances of tags from the interrogation area. Deterministic protocols, on the other hand, are ideally designed to identify the whole set of tags in the interrogation area during each cycle. These protocols usually have a simple tag design and can work very well with uniform sets of tags. However, these protocols do not admit unexpected appearances and disappearances as easily as Aloha-based protocols. Tree-based protocols must restart their reading process if a new tag appears in a reader's interrogation area while the tags are being read.

And finally, these types can be combined to form hybrid protocols, which provide very competitive protocols. Hybrid protocols have been created in order to avoid the problems of the Aloha and tree-based protocols, but this comes at the expense of complex reader and tag designs. Table 6 shows observations regarding Aloha, tree-based and hybrid protocols.

From these explanations of the protocols, it cannot be concluded that certain protocol types stand out from the rest. However it should be noted that the newest protocols have become more sophisticated and attain better results in simulations. This also contrasts with the ability to implement these solutions in real hardware. RFID systems are very constrained systems, and their hardware needs to be very simple in order to comply with tags' needs. That is why many of these solutions have yet to be tested under real hardware conditions.

Table 6. A comparison of Aloha, Tree-based and Hybrid protocols.

Criterion	Aloha Protocols	Tree-Based Protocols	Hybrid Protocols
Protocol feature	They use random multi-access means to identify tags. In the case of collision, tags will be asked to send data later with a random time relay.	They identify the total number of tags in the interrogation zone. The reader controls every step of the protocol, using commands or queries to split colliding tags into subsets, and further repeatedly split those subsets until identifying all of the tags.	Tree-based protocols. They use two methods. The first uses randomized divisions in tree-based algorithms, and another uses tree strategies after a collision in Aloha algorithms.
Number of tags to reader commands	Low	High	Medium
Usage	Aloha protocols are commonly used in LF, HF and UHF RFID (18000-6C) systems.	Tree-based protocols are commonly used in HF, UHF and microwave (18000-6B and 18000-7) RFID systems.	Not implemented on any standard
Method	Probabilistic	Deterministic	Mixture (Aloha and Tree-based)
Tag starvation	Yes	No	No

7. Conclusions

The term IoT, as established in RFID and the supply chain, involves the global information service architecture for RFID tags, that is, networked services that discuss things, rather than services. RFID is a

key opportunity for the IoT due to its cost-effectiveness, high readability rates, automatic identification and, importantly, its energy efficiency benefits.

This paper presents some of the main RFID procedures and proposes some of the most up-to-date anti-collision protocols. In the literature, these may be classified into Aloha-based, tree-based, and hybrid protocols. The breadth of the literature reveals that there has been considerable research carried out in this field. However, further research needs to be conducted in order to ultimately implement all these solutions. RFID systems have become more and more widely present. Given that the number of tags has increased, these system are faced with more important issues, and therefore, the use of anti-collision protocols will be more omnipresent.

Author Contributions: N.C. conceived the idea and wrote the paper. H.L. made valuable suggestions to analyze the data and improve the manuscript. N.C., H.L. and A.P. revised the manuscript.

Funding: This research was supported by the University of Deusto (Spain) and Ministry of Science (Montenegro).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. RFID-Journal. That Internet of Things. 2017. Available online: http://www.rfidjournal.com/articles/ (accessed on 15 April 2018).
2. Finkenzeller, K. *RFID Handbook*; Wiley: Hoboken, NJ, USA, 2010.
3. Azambuja, M.; Marcon, C.; Hessel, F. Survey of Standardized ISO 18000-6 RFID Anti-collision Protocols. In Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008), Cap Esterel, France, 25–31 August 2008; pp. 468–473.
4. Shih, D.; Sun, P.; Yen, D.; Huang, S. Taxonomy and survey of RFID anti-collision protocols. *Comput. Commun.* **2006**, *29*, 2150–2166. [CrossRef]
5. Klair, D.; Chin, K.; Raad, R. A Survey and Tutorial of RFID Anti-Collision Protocols. *IEEE Commun. Surv. Tutor.* **2010**, *12*, 400–421. [CrossRef]
6. Abraham, C.; Ahuja, V.; Ghosh, A.; Pakanati, P.P. *Inventory Management Using Passive RFID Tags: A Survey*; Department of Computer Science, The University of Texas at Dallas: Richardson, TX, USA, 2002; pp. 1–16.
7. Leong, K.; Ng, M.; Cole, P. The reader collision problem in RFID systems. In Proceedings of the 2005 IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, Beijing, China, 8–12 August 2005; pp. 658–661.
8. Burdet, L.A. RFID Multiple Access Methods. Smart Environments Seminar, Zurich. 2004. Available online: http://www.vs.inf.ethz.ch/edu/SS2004/DS/reports/06RFID-macreport.pdf (accessed on 20 April 2018).
9. Yu, J.; Liu, K.; Yan, G. A Novel RFID Anti-Collision Algorithm Based on SDMA. In Proceedings of the 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, Dalian, China, 12–17 October 2008; pp. 1–4.
10. Sayeed, M.; Kim, Y. A Simple LMS Algorithm Based Smart Antenna to Solve the Reader Collision Problems in RFID System. In Proceedings of the 2009 International Conference on Information and Multimedia Technology, Jeju Island, Korea, 16–18 December 2009; pp. 426–430.
11. Banks, J.; Pachano, M.; Thompson, L.; Hanny, D. *RFID Anti-Collision System Using the Spread Spectrum Technique*; Technical Report; Georgia Institute of Technology: Atlanta, GA, USA, 2005.
12. Liu, H.C.; Ciou, J. Performance analysis of multi-carrier RFID systems. In Proceedings of the 2009 International Symposium on Performance Evaluation of Computer Telecommunication Systems, Istanbul, Turkey, 13–16 July 2009; Volume 41, pp. 112–116.
13. Loeffler, A.; Schuh, F.; Gerhaeuser, H. Realization of a CDMA-based RFID System Using a Semi-active UHF Transponder. In Proceedings of the 2010 6th International Conference on Wireless and Mobile Communications, Valencia, Spain, 20–25 September 2010; pp. 5–10.
14. Tang, Z.; He, Y. Research of Multi-access and Anti-collision Protocols in RFID Systems. In Proceedings of the 2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID), Xiamen, China, 16–18 April 2007; pp. 377–380.

15. Wang, W.; Zhang, Y.; Sang, Y.; Wang, S. Analysis of anti-collision algorithms in RFID system. In Proceedings of the 2009 IEEE International Conference on Communications Technology and Applications, Boston, MA, USA, 16–18 October 2009; pp. 58–62.
16. Schoute, F. Dynamic Frame Length Aloha. *IEEE Trans. Commun.* **2010**, *31*, 565–568. [CrossRef]
17. Zhu, L.; Yum, T. The Optimal Reading Strategy for EPC Gen-2 RFID Anti-Collision Systems. *IEEE Trans. Commun.* **2010**, *58*, 2725–2733. [CrossRef]
18. Maguire, Y.; Pappu, R. An Optimal Q-Algorithm for the ISO 18000-6C RFID Protocol. *IEEE Trans. Autom. Sci. Eng.* **2009**, *6*, 16–24. [CrossRef]
19. Zhu, L.; Yum, T. Optimal Framed Aloha Based Anti-Collision Algorithms for RFID Systems. *IEEE Trans. Commun.* **2010**, *58*, 3583–3592. [CrossRef]
20. Chen, W.T. An Accurate Tag Estimate Method for Improving the Performance of an RFID Anticollision Algorithm Based on Dynamic Frame Length Aloha. *IEEE Trans. Autom. Sci. Eng.* **2009**, *6*, 9–15. [CrossRef]
21. Abramson, N. THE ALOHA SYSTEM: Another alternative for computer communications. In Proceedings of the Fall Joint Computer Conference, Houston, TX, USA, 17–19 November 1970; pp. 281–285.
22. Roberts, L. ALOHA packet system with and without slots and capture. *ACM SIGCOMM Comput. Commun. Rev.* **1975**, *5*, 28–42. [CrossRef]
23. Klair, D.; Chin, K.; Raad, R. An Investigation into thie Energy Eficiency of Pure and Slotted Aloha Based REID Anti-Collision Protocols. In Proceedings of the 2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Espoo, Finland, 18–21 June 2007; pp. 1–4.
24. Namboodir, V.; DeSilva, M.; Deegala, K.; Ramamoorthy, S. An extensive study of slotted Aloha-based RFID anti-collision protocols. *Comput. Commun.* **2012**, *35*, 1955–1966. [CrossRef]
25. Bueno-Delgado, M.; Vales-Alonso, J.; Gonzalez-Castaño, F. Analysis of DFSA anti-collision protocols in passive RFID environments. In Proceedings of the 2009 35th Annual Conference of IEEE Industrial Electronics, Pisa, Italy, 3–5 November 2009; pp. 2610–2617.
26. Vogt, H. Efficient object identification with passive RFID tags. In Proceedings of the 2002 International Conference on Pervasive Computing, Zürich, Switzerland, 26–28 August 2002.
27. Vogt, H. Multiple object identification with passive RFID tags. In Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics, Hammamet, Tunisia, 6–9 October 2002; Volume 3, pp. 1–6.
28. Cha, J.R.; Kim, J.H. Novel Anti-collision Algorithms for Fast Object Identification in RFID System. In Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05), Fuduoka, Japan, 22–25 July 2005; Volume 2, pp. 63–67.
29. Hwang, T.W.; Lee, B.G.; Kim, Y.S.; Suh, D.Y.; Kim, J.S. Improved Anti-collision Scheme for High Speed Identification in RFID System. In Proceedings of the First International Conference on Innovative Computing, Information and Control, Beijiang, China, 30 August–1 September 2006; Volume 2, pp. 449–452.
30. Wieselthier, J.; Ephremides, A.; Michaels, L. An exact analysis and performance evaluation of framed Aloha with capture. *IEEE Trans. Commun.* **1989**, *37*, 125–137. [CrossRef]
31. Yan, F.; Shao, X.; Sun, Q. The dynamic anti-collision algorithm based on the similar binary in RFID system. In Proceedings of the 2008 2nd International Conference on Anti-Counterfeiting, Security and Identification, Guiyang, China, 20–23 August 2008; pp. 444–447.
32. Lin, C.; Lin, F. Efficient Estimation and Collision-Group-Based Anti-collision Algorithms for Dynamic Frame-Slotted Aloha in RFID Networks. *IEEE Trans. Autom. Sci. Eng.* **2010**, *7*, 840–848. [CrossRef]
33. EPCglobal. *Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860–960 MHz, Version 1.2.0*; EPC EPCglobal: Beijing, China, 2008.
34. Wang, C.; Daneshmand, M.; Sohraby, K.; Li, B. Performance analysis of RFID Generation-2 protocol. *IEEE Trans. Wirel. Commun.* **2009**, *8*, 2592–2601. [CrossRef]
35. Lee, D.; Kim, K.; Lee, W. Q^+-Algorithm: An Enhanced RFID Tag Collision Arbitration Algorithm. In Proceedings of the 4th International Conference on Ubiquitous Intelligence and Computing (UIC 2007), Hong Kong, China, 11–13 July 2007; pp. 23–32.
36. Peeters, G.; Houdt, B. Interference Cancellation Tree Algorithms with k-Signal Memory Locations. *IEEE Trans. Commun.* **2010**, *58*, 3056–3061. [CrossRef]
37. Yeh, M.; Jiang, J.; Huang, S. Parallel Response Query Tree Splitting for RFID Tag Anti-collision. In Proceedings of the 2011 40th International Conference on Parallel Processing Workshops, Taipei, Taiwan, 13–16 September 2011; pp. 6–15.

38. Myung, J.; Lee, W.; Shih, T. An Adaptive Memoryless Protocol for RFID Tag Collision Arbitration. *IEEE Trans. Multimed.* **2006**, *8*, 1096–1101. [CrossRef]
39. Gou, H.; Jeong, H.; Yoo, Y. A Bit collision detection based Query Tree protocol for anti-collision in RFID system. In Proceedings of the 2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications, Chengdu, China, 23–25 September 2010; pp. 421–428.
40. Zein, W.; Shaaban, E. An enhanced binary tree anti-collision technique for dynamically added tags in RFID systems. In Proceedings of the 2010 2nd International Conference on Computer Engineering and Technology, Cengdu, China, 16–18 April 2010; Volume 7, pp. 349–353.
41. Law, C.; Lee, K.; Siu, K. Efficient memoryless protocol for tag identification. In Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Boston, MA, USA, 11 August 2000 .
42. Choi, J.; Lee, D.; Youn, Y.; Jeon, H.; Lee, H. Scanning-Based Pre-Processing for Enhanced Tag Anti-Collision Protocols. In Proceedings of the 2006 International Symposium on Communications and Information Technologies, Damascus, Syria, 24–28 April 2006; pp. 1207–1211.
43. Zhou, F.; Chen, C.; Jin, D.; Huang, C.; Min, H. Evaluating and Optimizing Power Consumption of Anti-Collision Protocols for Applications in RFID Systems. In Proceedings of the 2004 International Symposium on Low Power Electronics and Design, Newport Beach, CA, USA, 9–11 August 2004; pp. 357–362.
44. Choi, J.; Lee, D.; Lee, H. Query tree-based reservation for efficient RFID tag anti-collision. *IEEE Commun. Lett.* **2007**, *11*, 85–87. [CrossRef]
45. Bhandari, N.; Sahoo, A.; Iyer, S. Intelligent Query Tree (IQT) Protocol to Improve RFID Tag Read Efficiency. In Proceedings of the 9th International Conference on Information Technology, Orissa, India, 18–21 December 2006; pp. 46–51.
46. Pan, L.; Wu, H. Smart Trend-Traversal Protocol for RFID Tag Arbitration. *IEEE Trans. Wirel. Commun.* **2011**, *10*, 3565–3569. [CrossRef]
47. Yan, X.; Zhang, R.; Li, B. Smart Trend-Traversal Protocol with Shortcutting for Memoryless RFID Tag Collision Resolution. In Proceedings of the 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing, Fukuoka, Japan, 4–7 September 2012; pp. 857–862.
48. Landaluce, H.; Perallos, A.; Zuazola, I. A Fast RFID Identification Protocol with Low Tag Complexity. *IEEE Commun. Lett.* **2013**, *17*, 1704–1706. [CrossRef]
49. Landaluce, H.; Perallos, A.; Angulo, I. Managing the Number of Tag Bits Transmitted in a Bit-Tracking RFID Collision Resolution Protocol. *Sensors* **2014**, *14*, 1010–1027. [CrossRef] [PubMed]
50. Landaluce, H.; Perallos, A.; Onieva, E.; Arjona, L.; Bengtsson, L. An Energy and Identification Time Decreasing Procedure for Memoryless RFID Tag Anticollision Protocols. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 4234–4247. [CrossRef]
51. Cmiljanic, N.; Landaluce, H.; Perallos, A.; Arjona, L. Reducing Transmitted Bits in a Memoryless RFID Anti-Collision Protocol. *IEEE Commun. Lett.* **2016**, *14*, 713–715.
52. Cmiljanic, N.; Landaluce, H.; Perallos, A.; Arjona, L. Influence of the Distribution of Tag IDs on RFID Memoryless Anti-Collision Protocols. *Sensors* **2017**, *17*, 1891. [CrossRef] [PubMed]
53. Rohatgi, I.; Durgin, G. *RFID Applied*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
54. Jia, X.; Feng, Q.; Yu, L. Stability Analysis of an Efficient Anti-Collision Protocol for RFID Tag Identification. *IEEE Trans. Commun.* **2012**, *60*, 2285–2294. [CrossRef]
55. Wang, T.P. Enhanced binary search with cut-through operation for anti-collision in RFID systems. *IEEE Commun. Lett.* **2006**, *10*, 236–238. [CrossRef]
56. Yu, S.; Zhan, Y.; Wang, Z.; Tang, Z. Anti-collision algorithm based on jumping and dynamic searching and its analysis. *Comput. Eng.* **2005**, *31*, 19–20.
57. Jia, X.; Feng, Q.; Ma, C. An Efficient Anti-Collision Protocol for RFID Tag Identification. *IEEE Commun. Lett.* **2010**, *14*, 1014–1016. [CrossRef]
58. Lai, Y.; Hsiao, L.; Chen, H.; Lai, C.; Lin, J. A Novel Query Tree Protocol with Bit Tracking in RFID Tag Identification. *IEEE Trans. Mob. Comput.* **2013**, *12*, 2063–2075. [CrossRef]
59. Lai, Y.C.; Hsiao, L.Y.; Lin, B.S. Optimal Slot Assignment for Binary Tracking Tree Protocol in RFID Tag Identification. *IEEE/ACM Trans. Netw.* **2015**, *23*, 255–268. [CrossRef]

60. Hush, D.; Wood, C. Analysis of tree algorithms for RFID arbitration. In Proceedings of the 1998 IEEE International Symposium on Information Theory, Cambridge, MA, USA, 16–21 August 1998; pp. 107–125.
61. Zhang, L.; Zhang, J.; Tang, X. Assigned Tree Slotted Aloha RFID Tag Anti-Collision Protocols. *IEEE Trans. Wirel. Commun.* **2013**, *12*, 5493–5505. [CrossRef]
62. Bonuccelli, M.; Lonetti, F.; Martelli, F. Tree slotted aloha: A new protocol for tag identification in RFID networks. In *Instant Collision Resolution for Tag Identification in RFID Networks*; Elsevier: New York, NY, USA, 2007; pp. 1220–1232.
63. Qian, C.; Liu, Y.; Ngan, H.; Ni, L. ASAP: Scalable Identification and Counting for Contactless RFID Systems. In Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems, Genova, Italy, 21–25 June 2010; pp. 52–61.
64. Myung, J.; Lee, W.; Srivastava, J. Adaptive binary splitting for efficient RFID tag anticollision. *IEEE Commun. Lett.* **2006**, *10*, 144–146. [CrossRef]
65. Wu, H.; Zeng, Y.; Feng, J.; Gu, Y. Binary Tree Slotted Aloha for Passive RFID Tag Anticollision. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 19-31. [CrossRef]
66. Porta, T.L.; Maselli, G.; Petrioli, C. Anticollision Protocols for Single-Reader RFID Systems: Temporal Analysis and Optimization. *IEEE Trans. Mob. Comput.* **2011**, *10*, 267–279. [CrossRef]