Reducing Transmitted Bits in a Memoryless RFID Anti-collision Protocol

Nikola Cmiljanic, Hugo Landaluce, Asier Perallos and Laura Arjona
DeustoTech-Fundación Deusto, Deusto Foundation, Av. Universidades 24, 48007, Bilbao, Spain

Keywords: RFID, Anti-collision, Tag Identification, Window, Query Tree.

Abstract: The use of Radio Frequency Identification (RFID) technologies is growing. RFID enables data to be collected from many objects, for identification and other purposes. One of the main disadvantages in tag identification, known as the tag collision problem, is becoming significant, since it leads to the increase in the number of transmitted bits and identification time. The window methodology is created with the aim to manage the number of transmitted bits by the tags. As a result, tags transmit exclusively the bits defined by the window instead of sending their full ID value on every response. This paper presents a protocol with a standardized window. The window size is transmitted as the exponent of power of 2. Simulations show that the proposed solution with standarized window size reduces the number of bits transmitted by the reader, with respect to other protocols using the window, which results in a lower number of total bits in the identification process.

1 INTRODUCTION

Radio Frequency Identification (RFID) is very popular, and it is experiencing a higher growth every day. RFID is a technology that uses radio waves to automatically identify people or objects. This technology helps in many fields and provides information about people, goods and products in transport. As opposed to barcode, RFID does not require close handling, no line of sight is required between the reader and the object to be identified, and tags provide significantly greater storage capacity. Those are the reasons why RFID is increasingly present.

The main components of an RFID system are: a reader and transponders (in advance tags) (Finkenzeller, 2010). RFID uses radiofrequency waves in order to automatically identify and track tags attached to objects. The reader is the main component, which is in charge of managing the identification process and collecting data from the tags by using an antenna that broadcasts radio waves. A tag is a small label which is attached to an object that wants to be identified. The reader sends out electromagnetic waves that the tag antenna receives and backscatters with its ID, converting those waves into digital data.

Tags are categorized into: passive, active, and semi passive. Passive tags are smaller and do not have a power supply. They reflect the data received from the reader to transmit their data. Active tags have an integrated battery, which is used to provide voltage to the chip and to transmit a longer distance. Semi passive tags use a battery only to power on the chip, so the entire energy, which is received from the reader, can be used for the transmission of data. Tags are available in many shapes such as smart cards, coins, keys, key fobs, clocks, and smart labels. RFID systems can also be classified according to the working frequencies of the carrier signals: low frequency (LF) 125kHz-134kHz, high frequency (HF) 13.56Mhz, ultra high frequency (UHF) 860-930 MHz, and super high frequency (SHF). Depending on the frequency, tags have various achievable ranges, from a few cm to 8 m (Finkenzeller, 2010).

The process of tag identification comprises a reader interrogation and a tag response. One of the main problems in RFID systems happens when multiple tags are interrogated by the reader simultaneously and they reflect their respective signals back to the reader at the same time. As a result of this obstruction there is a failed transmission called collision. These collisions cause the reader to be unable to identify tags successfully and rapidly. Thus, RFID obtains low tag reading performance and a high waste of energy. To minimize the influence of the tag collision problem, RFID readers use an anti-collision protocol. Anti-collision protocols can be divided into three categories: Aloha-based protocols which are probabilistic, tree-based protocols which are deterministic, and hybrid protocols (Klair et al., 2010).

Aloha-based protocols (Wang et al., 2009; J. and T., 2008), come from computer networks and they use a random access strategy in order to identify tags.

Cmiljanic, N., Landaluce, H., Perallos, A. and Arjona, L.
Reducing Transmitted Bits in a Memoryless RFID Anti-collision Protocol.
DOI: 10.5220/0005963201520157
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 6: WINSYS*, pages 152-157
ISBN: 978-989-758-196-0

They are probabilistic protocols because tags send data in a randomly selected slot in a frame to reduce the possibility of collision. Accordingly, it is not guaranteed that all the tags will be identified (Wang et al., 2009). When a collision occurs the tag will be asked to send its data later with a random time delay.

Tree-based protocols are also known as deterministic protocols because all the tags in the interrogation zone will be read within certain time limits (Feng et al., 2006). A group of collided tags should be separated into subsets repeatedly until every tag in the interrogation zone responds correctly. The total amount of information transmitted between readers and tags is usually higher than in Aloha based protocols (Klair et al., 2010). Tree based protocols require tags to have 'sleeping' capability, as tags are silenced after the process of identification.

Hybrid protocols are created to avoid the problems of Aloha and tree-based protocols (Wu et al., 2013). Some advantages of hybrid protocols are that they can achieve a higher slot efficiency than the other types. A hybrid protocol consists principally of two combined protocols: one is using randomized divisions in tree-based algorithms, and another is using tree strategies after a collision in Aloha-based algorithms.

Tags in the proposed window methodology (Landaluce et al., 2013) can not locate the bit string representing the window size in the received command. In this paper, the window standardization is performed. The window is represented in a fixed size string of 3 bits. When tags receive a command, they can separate the part of the reader query from the part of the window size string.

The rest of the paper is organized as follows. Section 2 presents the window methodology and describes three tree based protocols of the state of the art. In section 3 the proposed Standardized Query Window tree protocol (SQwT) is presented. Section 4 presents evaluation results and the comparison with some state of the art protocols. Finally, Section 5 concludes the whole paper.

2 RELATED WORK

Some basic terms are given before presenting the most related tree based protocols:

- Query is a broadcast command sent by the reader. This command consists of a prefix (binary string) that the tags will compare with their ID. If tags' ID does do not match the query, the reader command will be rejected. In case that tags' ID matches the query, a response will be transmitted according to each protocol.
- Slot is the time interval which is intended to organize the responses from the tags. Three types of slots can occur depending on the number of tag responses received at the reader: idle slot, when (upon reader request) no tag answers; collision slot, when more than one tag respond to a reader query, and the reader can not understand the response; and success slot, when the reader successfully identifies a tag.
- Identification process is the period including all the reader queries and tags responses. It consists of a certain number of time slots needed by the reader to identify the entire set of tags.

Here some of the most important tree based protocols are presented.

2.1 Query Tree Protocol

The Query Tree protocol (QT) is one of the most significant tree based protocols (C. Law and Siu, 2000). It is called memoryless protocol and it means that a tag response depends on the current query but not on the past history of the reader queries. QT tags have the lowest hardware requirements because they only compute a prefix-comparison operation between a query and the tag ID. In each round the reader transmits a query and the tags whose ID matches it, will respond. In case that more than one tag respond, a collision occurs and the reader will create two new queries appending values 1 and 0 to the query. That is, two new queries will be placed in a Last Imput First Output stack (LIFO). If there is no answer upon a query command, the reader knows that there is not any tag with the required prefix and the query is rejected. When a prefix matches a tag's ID, the tag transmits the complete ID. By extending the prefixes until only one tag's ID matches, the algorithm can identify the rest of the tags. The identification process is completed when the stack is empty. QT tags transmit the whole ID every time they match a query. In case more than one tag responds, a collision occurs. Accordingly, all tags that match the query prefix transmit their full ID and a lot of bits are wasted upon every collision.

2.2 Smart Tree Traversal Protocol

Smart Tree Traversal protocol (STT) (Pan and Wu, 2011) is created in order to reduce the number of collisions in QT. STT is a tree based protocol with the ability of self-learning. The reader in this protocol dynamically generates queries according to the ability to recognize tag density and distribution. Depending on the result of the tag response, the reader calculates the next query:

- Upon a collision, the reader appends x bits of 0's on the last transmitted query where x=r+w_{col}-1, r denotes the minimum increase, and w_{col} is the number of consecutive collisions slots.
- When an idle slot occurs, the reader decreases the query length by the number of bits y, where y=r+w_{emp}-1, by which to decrease the query, and w_{emp} is the number of consecutive idles.
- Upon a success response, the reader visits the symetric node if the query finishes with 0 or it returns one level if it finishes with 1.

On every collision, the full tag response, apart from the initial query bits, is wasted. In order to alleviate this issue, the window methodology is presented.

2.3 Query Window Tree Protocol

Instead of transmitting the whole ID value from the tags, every tag response is limited. The Query window Tree protocol (QwT) is presented in the literature (Landaluce et al., 2013). This methodology is applied to the QT protocol to control the number of transmitted bits by tags. The number of collisions is decreased by transforming potential collisions into partial successes called Go-On slots. The window is a bit-string transmitted by the tags, instead of their full ID. If tags match a reader query, they will synchronously transmit the amount of bits of the ID specified by the window size, *ws*, instead of their full ID. This is shown in Fig. 1. QwT uses cyclic redundancy check (CRC) to differentiate between the type of tags' responses, which can be classified into four types:

- Idle slot: when no tag responds upon a reader query. The reader rejects this query and continues with the identification process by sending the next query from the stack.
- Collision slot: when more than one tag responds the window to the query, but the CRC received by the reader is not consistent. The reader creates two new queries: $[q_1, q_2 \dots q_L, 0]$ and $[q_1, q_2 \dots q_L, 1]$ where $q_i \in [0,1]$. The reader transmits the first generated query with unchanged *ws* and stores the second one into the stack.
- Go-On slot: when at least one tag responds the window and the reader can understand it. If the ID is not complete L+ws<k, where L is the length of the query and k the length of the tag ID. It is assumed a Go-On slot and the reader continues to interrogate tags with an updated query made from the last received window and the former query.

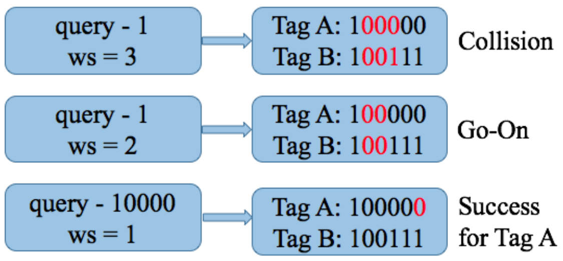

Figure 1: Window synchronized answer.

- Success slot: when the reader successfully receives the remaining values of the ID. The reader collects the complete ID and stores it in a database. In this slot L+ws=k is met.

Using the proposed QwT, the reader in each interrogation round transmits a query of L bits and preselects *ws* using the expression (1), where β is an adjustable parameter (Landaluce et al., 2013).

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

When the calculated ws is high, the reader command needs a high number of bits to represent it. That leads to a wastage of the reader bits. In order to reduce the number of the reader bits this paper proposes the standardization of the window methodology.

3 STANDARDIZED QUERY WINDOW TREE PROTOCOL

The presented protocol proposes the standardized window to restrict the number of bits transmitted by the reader, reducing the total number of transmitted bits. A decrease of the reader bits is achieved by reducing the amount of the window size bits in the reader command. The presented protocol has the ability to locate the bit string representing the window size in the received command, differentiating it from the query by using a fixed size at the end of the reader command.

SQwT is also a memoryless protocol since the current response of each tag only depends on the current query and *ws*. Tags that match the reader prefix will exclusively transmit *ws* bits.

A heuristic function which computes *ws* is given in (1). An adequate value of β is 0.5 for the lowest Go-On slots rate (Landaluce et al., 2013). After calculating *ws*, the reader will choose the first higher power of 2 value. The obtained value is the number of bits that matching tags should transmit from the ID.

The reader calculates the number of bits that tags must respond to a matching query. Instead of sending

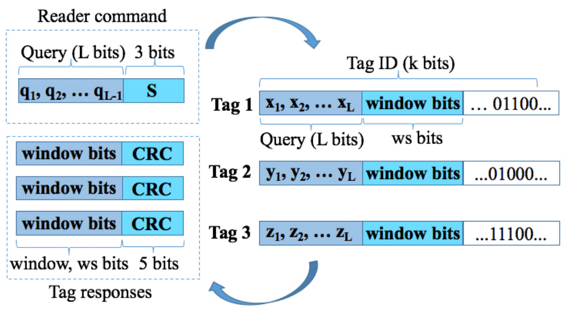

Figure 2: Format of the reader command and tag responses of the SQwT protocol.

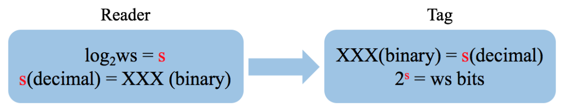

Figure 3: Converting WS between reader and tag.

ws bits, SQwT will transmit a new value *s* which is presented in (2).

$$s = log_2ws \tag{2}$$

As shown in Fig.2, the reader interrogates tags with a query $[q_1, q_2 \dots q_L]$ and a fixed string *s*. All tags will receive this broadcast message but only tags whose ID matches the query will calculate *s* and respond to the reader request. SQwT gives a constant place of 3 bits included in the reader command. It allows tags to locate the amount of the window size string in the received command. The matching tags calculate *ws* by using the expression (3) and get the final value.

$$ws = 2^s \tag{3}$$

The process is depicted in Fig.3. E.g., if *ws*=128, the reader, first uses (2) with result 7. This is codified into the binary number 111. The reader transmits this bit string attached to the predefined query. The ID length of 128 bits is the most common in RFID (GS1 and EPCglobal,). For this reason SQwT uses 3 bits in the reader query for *s*. For longer IDs, a larger value of *s* should be used, and a longer bit string would be needed.

The reader and tag flow chart of SQwT are given in Fig 4.(a) and (b). First, the reader initializes by pushing two queries into a LIFO stack and the reader pops the last pushed query. For the initial *ws* calculation, it uses *ws*=1. In other rounds, *ws* should be calculated using (1) and the nearest power of 2 should be chosen towards ∞. Subsequently, the reader obtains *s* with (2) and codifies it with 3 bits. Once *s* is calculated, a new reader command including a query and *s* will be transmitted to the tags. The reader awaits for the tags' answers. Tags receive the reader command (see Fig. 4.b) and compare the query with their ID. Matching tags will calculate *ws* using (3) and will respond the remaining ws bits from bit L $w_1 \dots w_{ws}$, and a CRC. Depending on the type of response, the reader will act as follows:

- Upon a collision the reader creates two new queries: $q_1, q_2 \dots q_L, 1$ and $q_1, q_2 \dots q_L, 0$ and *ws* is unchanged.
- For the cases when an empty window is received, the transmitted query is rejected and another one query is popped from the stack. The windows size remains unchanged.
- In case the received CRC is positively checked, the reader checks the expression *L*+*ws*=*k*, and if it matches, the tag is successfully identified. A new query is popped from the stack and textitws is calculated using (1).
- In case detecting a Go-On (*L*+*ws*<*k*) slot, the reader produces a new query attaching $w_1, w_2 \dots w_{ws}$ to the last transmitted query until it receives the full ID value. The windows size is calculated using (1).

This procedure sequences repeatedly until it gets the empty stack.

An example of identification of six tags is depicted in Fig.5 using QT and the proposed SQwT protocol. The ID length *k* used is 8 bits and *ws* is computed using exponential heuristic function with β=0.5. The reader first sends the query (0). Tags A, B, C and D respond with the whole ID value and a collision occurs using QT protocol. The difference with SQwT protocol is in the tag response. After matching query (0) the same tags as in the QT protocol will transmit but just with the first bits, defined by *ws* value. The reader can receive a partially successful response add it adds the bit '0' to the previous query forming

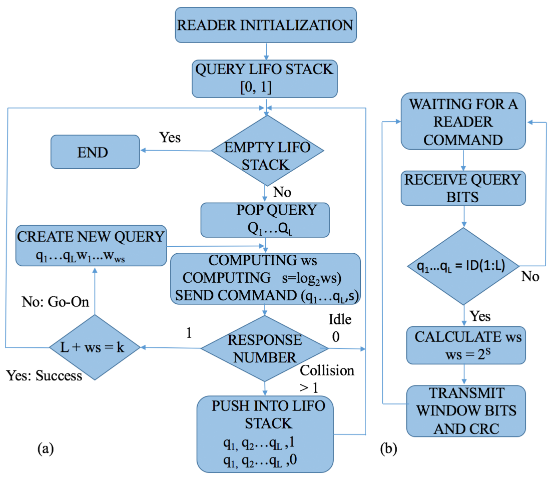

Figure 4: Flow chart of the proposed SQwT protocol: (a) for reader, (b) for tags.

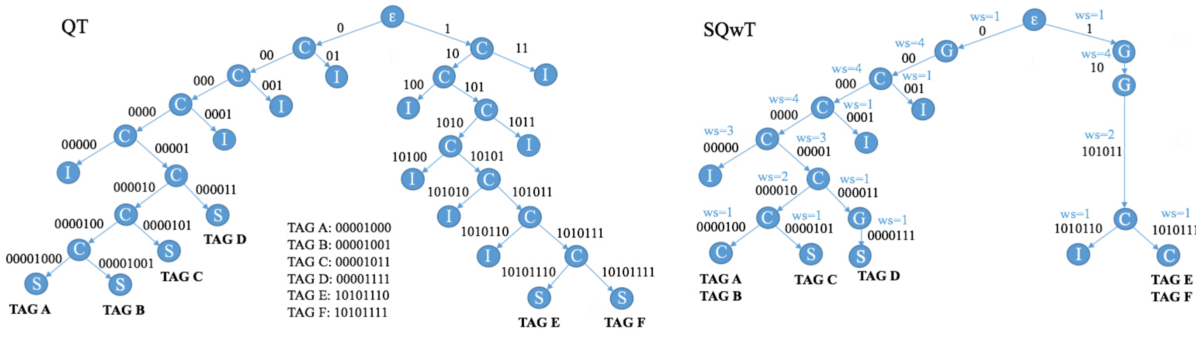

Figure 5: Example of the proposed SQwT and the QT protocol.

a new query (00) and it continues with the identification proccess. In this case (1) calculates *ws* 4 bits. Again four tags match the query (00) and transmit the following 4 bits. In this round, the reader can not understand the responses and collision occurs. The following part of tree is the same in both protocols because 3 collided slots will occur. Every time a collision occurs the reader pushes into the stack two new queries: $[q_1, q_2 \ldots q_L, 0]$ and $[q_1, q_2 \ldots q_L, 1]$. The left subtree under the 0 branch of both protocols is the same untill transmitting query 0000100. QT protocol needs to make one more interrogation round because tags respond the whole ID values as opposed to SQwT which transmits just the last bits, and tags A and B can be identified. After this success slot, the reader pops the next query and can identify Tag C. The reader pops out a new query from the stack (000011), resulting in the identification of tag D. QT protocol needs two more slots in order to identify the same number of tags. The right part of the tree is significantly reduced in the SQwT protocol. After transmitting the first query '1' from the right side of the tree, two Go-On slots will occur. Afterwards, SQwT pops the next query (101011) and detects a collision. The last query created during the last collision results in the identification of the remaining two tags: Tag E and Tag F. Eventually, when the comparison of QT and SQwT is analyzed, results show a lower number of transmitted bits per tag, and a decrease of the number of collisions.

4 EXPERIMENTATION

This section presents the evaluation of the simulation results of the presented SQwT with QT (C. Law and Siu, 2000), QwT (Landaluce et al., 2013) and STT (Pan and Wu, 2011). The simulations are executed using MatLab R2014. The number of tags has been varied from 100 to 1000. Simulations are averaged over 100 times for accuracy in the results, and the IDs are randomly generated for every iteration with a length of 128 bits. CRC is assumed 5 bits. Table 1 shows the calculation of the number of transmitted bits for the protocols used in the simulation.

Simulated results in Fig.6.a present the reader bits used in the identification process. Presented results show evidence of decreased reader bits for the proposed SQwT protocol in the comparison with window based protocol, especially in dense tag environments. SQwT outperformed QwT in reader bits and consumes total bits similar to STT. The total number

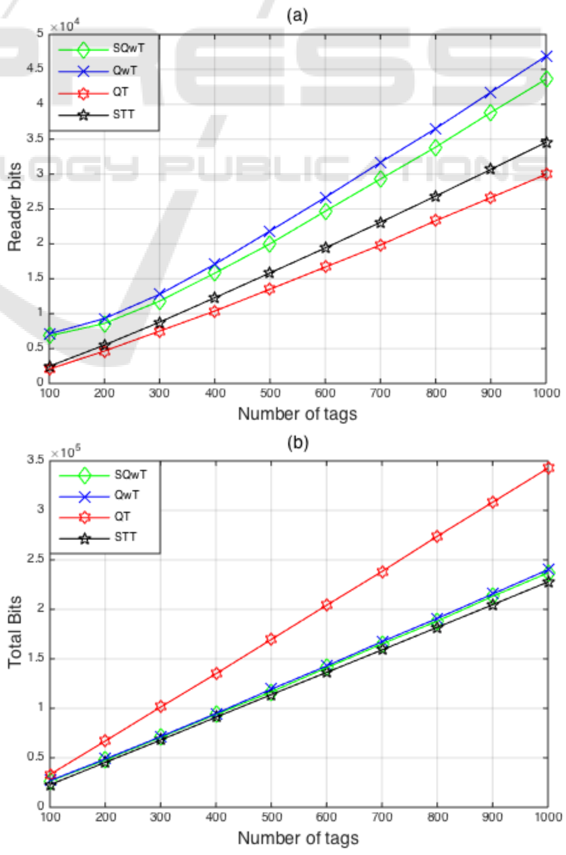

Figure 6: Performance of the SQwT: (a) reader bits, (b) total bits.

Table 1: Calculation of transmitted bits used in simulation.

Protocol	Reader command	Tag response
SQwT	$L+3$	$ws+CRC$
QwT	$L+\lfloor\log_2 ws\rfloor+1$	$ws+CRC$
QT	L	k
STT	L	$k-L$

of bits transmitted between the reader and the tags is shown in Fig.6.b. It is calculated as the number of transmitted bits by the reader plus the number of tag bits received on the reader side. Results indicate that SQwT significantly outperformed QT in total bits with the influence of the window methodology. The results indicate that SQwT is a dexterity protocol, which using a standardized window, reduces the number of transmitted reader bits used in the identification process.

5 CONCLUSIONS AND FUTURE WORK

A novel SQwT protocol for reducing the number of reader bits is presented here. The standardized window methodology is proposed and the window size is represented as a fixed string of 3 bits. The basic approach is the tag ability to locate the bit string representing the window size in the received command, differentiating it from the query. It is achieved by giving a constant place of 3 bits included in the reader command. Simulated results proved the dexterity of SQwT and outperformed other window based solution since the number of transmitted reader bits are significantly reduced. Future work will look at the possibility of implementing SQwT to be adapted to different ID distributions.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish Ministry of Economy and Competitiveness under the iLogisTICs project (TEC2013-45585-C2-2-R) and the Basque Government under the UBITRACE project (PC2013-71A)

REFERENCES

C. Law, K. L. and Siu, K. (2000). Efficient memoryless protocol for tag identification. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications,*.

Feng, B., Li, J.-T., Guo, J.-B., and Ding, Z.-H. (2006). Id-binary tree stack anticollision algorithm for RFID. In *11th IEEE Symposium on Computers and Communications, 2006. ISCC '06. Proceedings.*, pages 207–212.

Finkenzeller, K. (2010). *RFID Handbook*. Wiley.

GS1 and EPCglobal. "GS1 EPC Tag Data Standard 1.6" GS1, EPCglobal, Sept. 2011.

J., E. and T., L. (2008). Crc-16-based collision resolution in EPCglobal Class1 Generation2 RFID systems. In *Proceedings of the International Conference on Wireless Information Networks and Systems*, pages 61–65.

Klair, D., Chin, K.-W., and Raad, R. (2010). A survey and tutorial of RFID anti-collision protocols. *Communications Surveys Tutorials, IEEE*, 12(3):400–421.

Landaluce, H., Perallos, A., and Zuazola, I. (2013). A fast RFID identification protocol with low tag complexity. *IEEE Communications Letters*, 17(9):1704–1706.

Pan, L. and Wu, H. (2011). Smart trend-traversal protocol for RFID tag arbitration. *IEEE Transactions on Wireless Communications*, 10(11):3565–3569.

Wang, C., Daneshmand, M., Sohraby, K., and Li, B. (2009). Performance analysis of RFID Generation-2 protocol. *IEEE Transactions on Wireless Communications*, 8(5):2592–2601.

Wu, H., Zeng, Y., Feng, J., and Gu, Y. (2013). Binary tree slotted aloha for passive RFID tag anticollision. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):19–31.