14

RFID Model for Simulating Framed Slotted ALOHA Based Anti-Collision Protocol for Muti-Tag Identification

Zornitza Prodanoff[1] and Seungnam Kang[2]
[1]*University of North Florida*
[2]*National Seoul University*
[1]*USA*
[2]*South Korea*

1. Introduction

Radio Frequency Identification (RFID) networks use radio signal broadcast to automatically identify items with attached RFID tags. A tag consists of a microchip that stores a unique identifier and an antenna. The tag's antenna is attached to the chip and can transmit a unique tag identifier to a reader (also called interrogator). The reader is capable of learning the set of tags within its interrogation range. The process of learning in-range tags is called a census. After an initial census is completed, the reader can answer queries about the presence of specific tag(s) within its range sent to it from other type of devices.

RFID systems have abundant benefits as compared to the barcode and smart card systems. RFID networks use radio frequency as a method of data transmission. Thus, unlike barcode labels, a tag does not need to be placed in a line of sight position from the reader, or even get in contact with a reader as smart cards, in order to be identified successfully. Depending on whether they use low, high, or ultrahigh transmission frequencies, RFID tags are identifiable within 3 meters span in case of a typical far-field reader [Want06] or at even further distances. Therefore, RFID tags are used more flexibly and conveniently than existing barcode and smart card implementations.

Moreover, some commercial implementations of RFID tags can store data in the amount of 16bytes - 64Kbytes [Finkenzeller03]. RFID tags can hold the same amount of data compared to smart cards, and much larger volume than barcodes. In addition, RFID tags are getting less expensive. The cost of RFID chips at the time of this study is less than 10 cents, while back in 1999, for example, was around 2 US dollars. Since tag readers have limits on their operations range imposed by the frequency of the wireless signal used, when RFID networks need to cover large spaces, multiple readers need to be used. The cost of current reader implementations is hundreds of US dollars. As a result, RFID networks may not be yet suitable to track large inventories of inexpensive items, but they are certainly becoming more affordable and can be used to track different types of items, e.g. live stock, pets, and valuable goods. Due to these advantages RFID systems are emerging as one of the alternative technologies of our time.

One of the world biggest supply chains Wal-Mart has required suppliers to implement RFID networks in at least 12 of its 137 distribution centres by the end of 2006. The Proctor & Gamble Co. is the first of about 100 suppliers to conform to Wal-Mart's requirements to tag its products with RFID chips [Computerworld07]. The US Navy finished its pilot of a passive RFID system to support the loading of supplies into cargo containers in May 2004. According to the related final report the RFID process increased the speed and efficiency of the cargo checking process, while less people were needed to support the new RFID based system as compared to the legacy implementation [Weinstein05].

1.1 Physical composition

An RFID system is made up of an application, a reader and tags.

- The application is a program installed on a (proxy) computer which can control readers.
- The reader is a device which runs functions such as reading, writing and authentication. When the reader gathers data from tags it transmits to the computer application.
- The tag is used to identify an object and is located on (or in) the object itself.

A reader is connected to the computer and has a transmitter and receiver, while a tag has a control unit (chip) and a coupling element (antenna).

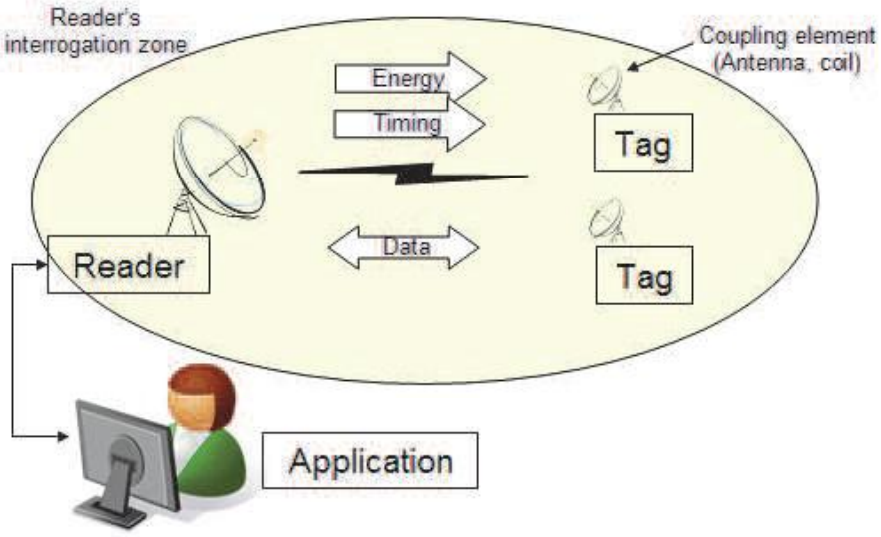


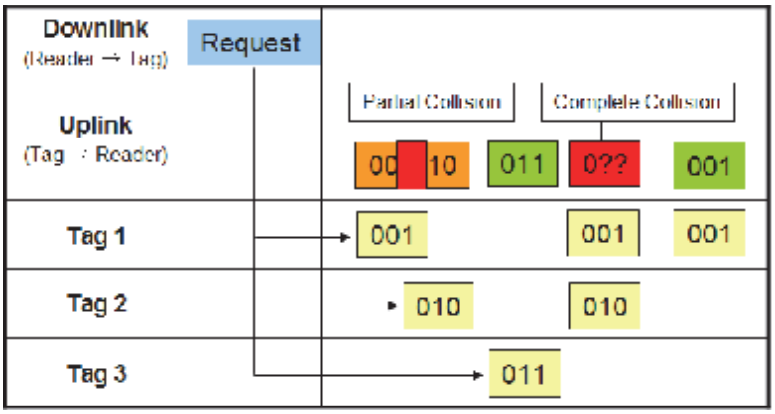
Fig. 1. RFID Physical Composition [Finkenzeller03]

RFID tags can be *passive*, i.e. not having an internal energy source or *active*, internal battery powered. A reader typically charges a set of passive tags within its interrogation zone using *inductive coupling*; the reader broadcasts electromagnetic signal then the tag's antenna absorbs and stores the signal's energy into an on-board capacitor. This technique is called *load modulation* for near-field coupling and *back scattering* for far-field coupling. After charging its battery it can be activated.

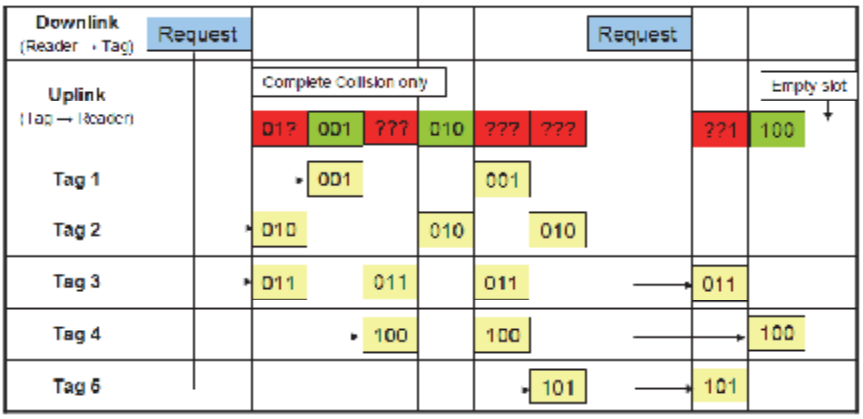
1.2 Framed slotted ALOHA anti-collision algorithm

The ALOHA algorithm is a collision resolution algorithm based on Time Division Multiple Access (TDMA). There are three flavors of the original ALOHA algorithm: (Pure) ALOHA, Slotted ALOHA and Frame Slotted ALOHA [Zürich04].

In Figure 2, X and Y axis represents the read cycle and tags respectively. The read cycle is the time interval between neighboring two *REQUEST* commands and it can be repeated until all tags in the interrogation range are identified. Note that there no slots are used in the (Pure) ALOHA algorithm (Figure 2: (a)) while the read cycle is divided into several continuous slots in the Slotted ALOHA (Figure 2: (b)) and Framed Slotted ALOHA algorithm. Furthermore, a frame is comprised of the number of slots in the Framed Slotted ALOHA algorithm (Figure 3: A *slot* is a discrete time intervals synchronized by the reader, sufficiently long in duration to allow a tag to transmit its ID and the ID's 16-bit CRC code. A set of slots are grouped into *frames*. When size is fixed, each consecutively transmitted frame has the same number of slots.



(a) (Pure) ALOHA



(b) Slotted ALOHA

Fig. 2. Pure and Slotted ALOHA Algorithms

The reader broadcasts the *REQUEST* command to the tags located in the reader's interrogation range during the downlink while the tags transmit their data to the reader during the uplink. As all activated tags share the uplink partial or complete collision can occur in the (Pure) ALOHA algorithm. However, if the data is transmitted using the slot of frame the partial collision can be eliminated. Furthermore, to reduce the fraction of collision occurrence tags send their data no more than once within a frame, which is the Frame Slotted ALOHA algorithm. We next present in more detail the operation of the three ALOHA algorithms introduced above.

1.2.1 (Pure) ALOHA

A tag itself decides the data transmission time randomly as soon as it is activated. The transmission time is not synchronized with both the reader and the other tags at all. When the electricity is charged by the reader's electromagnetic wave tags transmit data after receiving the *REQUEST* command from the reader. If multiple tags transmit data imminently (whether earlier or later) then a complete or partial collision occurs (Fig. 2 (a)). Retransmitting after random delay is the solution for a collision. During the read cycle the reader receives the data and identifies tags sent data without collision. When a read cycle is done then the reader broadcasts the *SELECT* command with the tag's unique identifier received from the tag. Once tags are selected the tags stop responding for the request command i.e. the selected tags keep silence until whether they receive other commands e.g. authenticate, read and write or the tag's power is off by being located out of the reader's power range. When the tag is reentered into the reader's interrogation range it restart transmitting its data to the reader. The advantage of this algorithm is simplicity.

1.2.2 Slotted ALOHA

It is obtained by the addition of a constraint to the (Pure) ALOHA. The read cycle is divided into discrete time intervals called *slot* and which is synchronized with the entire tags by the reader. Thus, tags must choose one of the slots randomly and transmit data within a single slot. Transmission begins right after a slot delimiter (Fig. 2 (b)). This causes that packets either collide completely or don't collide at all i.e. there is no partial collision in the Slotted ALOHA algorithm. This reduces wasting the read cycle relatively as compared with the (Pure) ALOHA algorithm. However, the empty slot can be occurred in the read cycle and the disadvantage is that it requires a synchronization mechanism in order for the slot-begin to occur simultaneously at all tags.

1.2.3 Framed slotted ALOHA

Framed Slotted ALOHA algorithm uses the frame which is the discrete time interval of the read cycle and each frame is divided into the same number of slots. There are multiple frames in a single read cycle and the frame size is decided by the reader (Figure 3: There is a constraint that the tags can transmit data only once in each frame. It may reduce the number of collided slots and it shows the best performance among them.

1.3 Classification of the framed slotted ALOHA protocol

FSA (Framed Slotted Aloha) can be classified into the BFSA (Basic Framed Slotted Aloha) and the DFSA (Dynamic Framed Slotted Aloha) according to whether which uses fixed frame size or variable frame size [Klair04]. If the number of actual tags is unknown DFSA

can identify tags efficiently rather than BFSA by changing frame size since BFSA uses fixed frame size. In addition, BFSA and DFSA can be further classified based on whether they support muting or/and early-end features [Klair04]. The muting makes tags remain silent after being identified by the reader while the early-end allows a reader close an idle slot early when no response is detected. Figure 4 is shown for the classification of the FSA.

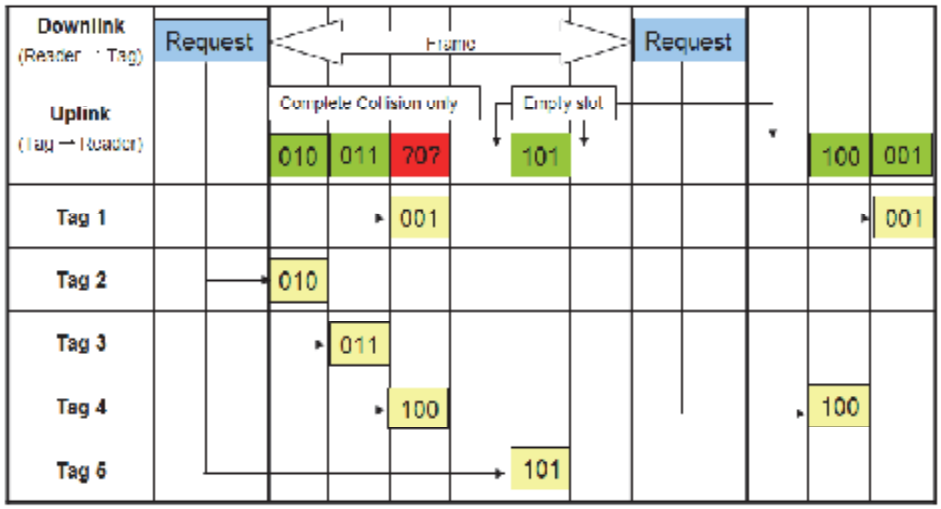


Fig. 3. Framed Slotted ALOHA Algorithms

1) BFSA
- BFSA-Non Muting
- BFSA-Muting
- BFSA-Non muting-early-end
- BFSA-Muting-early-end

2) DFSA
- DFSA-Non Muting
- DFSA-Muting
- DFSA-Non muting-early-end
- DFSA-Muting-early-end

Fig. 4. Classification of FSA

2. RFID network protocol simulation using OPNET

Framed Slotted ALOHA and Binary Tree are the two most widely used multi tags identifying anti-collision protocols. Fabio Cappelletti et al. simulated the Binary Tree protocol of RFID by using the OPNET IT Guru 11.0 in 2005 [Cappelletti06]. In the paper, they measured the network throughput and the census delay through the simulation. And they compared simulation performance and analytical results. What they measure is shown in Figure 5.

Analytical Parameters	Simulation parameters	Unit
Network throughput	Network throughput	(%)
Throughput per node	Throughput per node	(%)
Total census delay Lower bound Upper bound Arithmetic average Heuristic	Total census delay	Number of slots
Time required to detect a single tag Lower bound Upper bound Arithmetic average Heuristic	Time required to detect a single tag	Number of slots
Number of transmitted packets (Total, Average)		Number of packets

Fig. 5. The Measured Parameters for Analysis and Simulation

The network throughput represents the ratio between the number of successfully transmitted packets and the total number of packets sent by the tags while the throughput per node denotes the average number of packets sent by a single tag. The results of the paper showed that the analytical performance was in good agreement with the simulation results.

2.1 Framed slotted ALOHA protocol performance evaluation using Philips I-Code system

2.1.1 The I-Code RFID system

It is commercial product of RFID system which is comprised of the actual tags and a reader connected to the computer. The application is installed in the computer to control the reader collecting data from tags. The built-in Framed Slotted ALOHA protocol is provided by the system. The memory size of an I-Code tag is the total of 64 bytes which are available for 46 bytes application data, 8 bytes serial number and 10 bytes functionalities such as write protection, maintaining quiet state of tag and reset quite state, etc. And, the reader provides the interface for setting configuration parameters such as the serial connection speed and commands for handling communication with tags. Examples of commands are following:

- *Anti-collision/select (ACS)*: After broadcasting this command, tags begins transmitting their serial numbers. Once tags become "selected" status then remain quite in following ACS commands.
- *Read*: This command makes the selected tags transmit their data to the reader.
- *Write*: This command makes the selected tags write data transmitted by the reader.

2.2.2 The procedure of tag identification

The procedure of tag identification is different based on the classification of RFID [Finkenzeller03]. There are two types of procedure based on the tags characteristic in the Frame Slotted ALOHA protocol. One of them switches off when read while the other not switches off but replays transmission. The I-Code system uses the switching off tag. Figure 6 depicts the census procedure used in the I-Code System implementation.

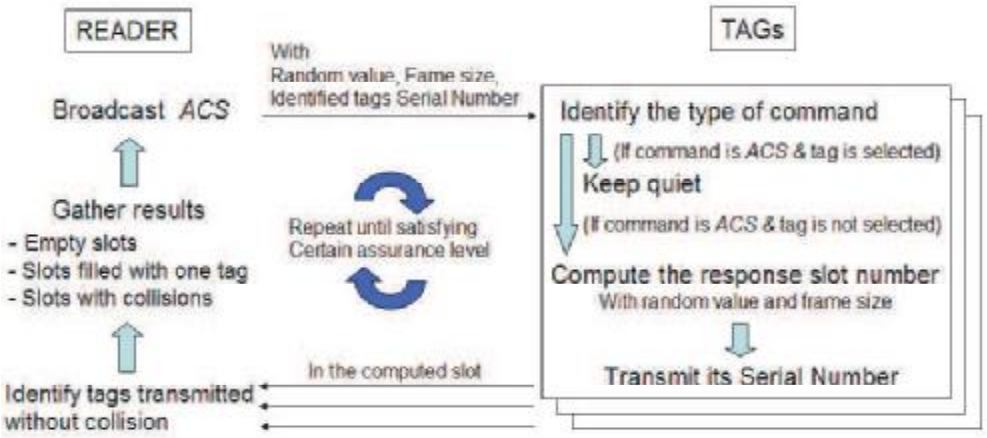


Fig. 6. Tag Identification Procedure of the I-Code System

The reader broadcasts the *ACS* command with a random value, frame size to make tags transmit their serial number to the reader. When the tag receives a command from the reader first of all it identifies the type of command, and if it is *ACS* and the status of tag is not 'selected' then it computes the response slot number using the random value and frame size ($0 \leq$ response slot $<$ frame size) as parameters. Random value is used to prevent the same collisions from occurring repeatedly. The serial number in the computed slot is transmitted to the reader by the tag. During an uplink, data transmission from tags to reader, the reader can identify tags transmitted without collision. The results of a read cycle (the number of empty slots, the number of slots filled with one tag and the number of collided slots) are used for analysis. Once the tag is identified the tag remains quiet until other command e.g. read or write is broadcasted with serial number. And if the tag re-enters the reader's power zone after moving out, that needs re-identification. The read cycle is repeated till reaching the assurance level, the probability of identifying all tags in the reader's range.

2.2.3 Basic framed slotted ALOHA

Through simulation we can measure the total census delay by varying the frame size for given number of tags. Then we can find the optimal frame size which results in the minimum total census delay, for given number of tags (see Figure 7).

The optimal frame size, resulting in minimum census delay, can be determined according to the total number of tags. Figure 8 presents an example of the relationship between total number of tags and frame size. For example, the optimal frame size for 80 active tags is 45, while for 30 passive tags it is 40.

2.2.4 Dynamic slot allocation for dynamic framed slotted ALOHA

To maximize network throughput frame size (the number of allocated slots in the read cycle) should be chosen in accordance with the number of tags since for the same fixed slot size, number of total collisions during a census increases with increase in total number of tags.

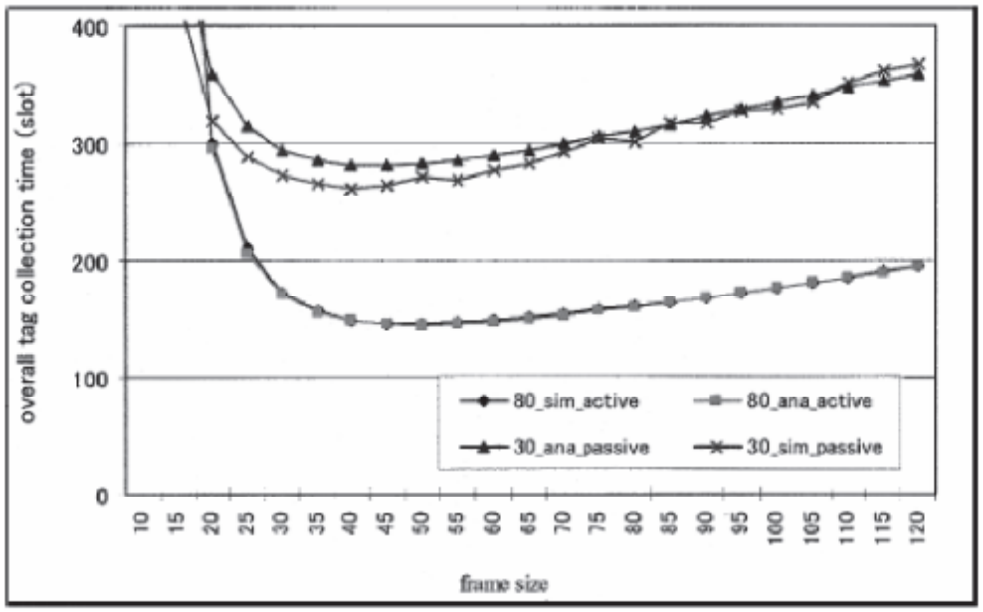


Fig. 7. Example of Total Census Delay (Tag Collection Time) Using Static Frame Size [Bin05]

Due to the nature of the Framed Slotted ALOHA protocol, the read cycle time is divided by the number of slots in a frame and packet data (tag ID number plus CRC) to be transmitted should occupy a single slot. If there are lots of tags and frame size is small then the probability of collisions will be increased and the number of identified tags is decreased, because tags will be competing for a lesser number of slots within a frame. On the contrary, if the reader reads few tags with too big frame size then the probability of collision is decreased , but at the expense of the response time being increased. There is optimal frame size that makes minimum number of read cycles for certain number of tags. The dynamic slot allocation is choosing the optimal frame size (in number of fixed slots) during the tag read cycle. However, the problem is that the number of tags is unknown. So, the reader should estimate the number of tags in each read cycle with the result of the previous read cycle (number of empty slots, number of slots filled with one tag, number of collided slots) and current frame size.

2.3 Developed approaches of the framed slotted ALOHA protocol

The key difference of the developed Framed Slotted ALOHA protocol is how they estimate the number of tags and what they estimate.

2.3.1 H. Vogt's algorithm

In this scheme, they estimate the number of tags using the current frame size and the result of read cycle. And then updates the current frame size using the estimated number of tags and previous frame size. The procedure of this algorithm is shown in Figure 9. Variable 'N', 'N0', 'n_est' and 'stepN' represent the frame size, temporary frame size, the

estimated number of tags and the counter for the cycle performed with currently estimated framed size, respectively. Variable 'c', 't' represents the result of a read cycle comprised of three integers; number of empty slots, number of slots filled with one tag and number of collided slots, and variable 't' represents the temporary number of estimated number of tags.
In order to estimate the number of tags two estimation functions are used; lower bound and Chebyshev's inequality. Lower bound simply estimates the number of tags is bigger than the summation of the number of slots filled with one tag and two-times of the number of collided slots:

$$\text{Number of tags} \geq \text{Number of filled slots with one tag} + 2 \times \text{Number of collided slots} \quad (1)$$

When the lower bound is used the real value of number of tags is underestimated.

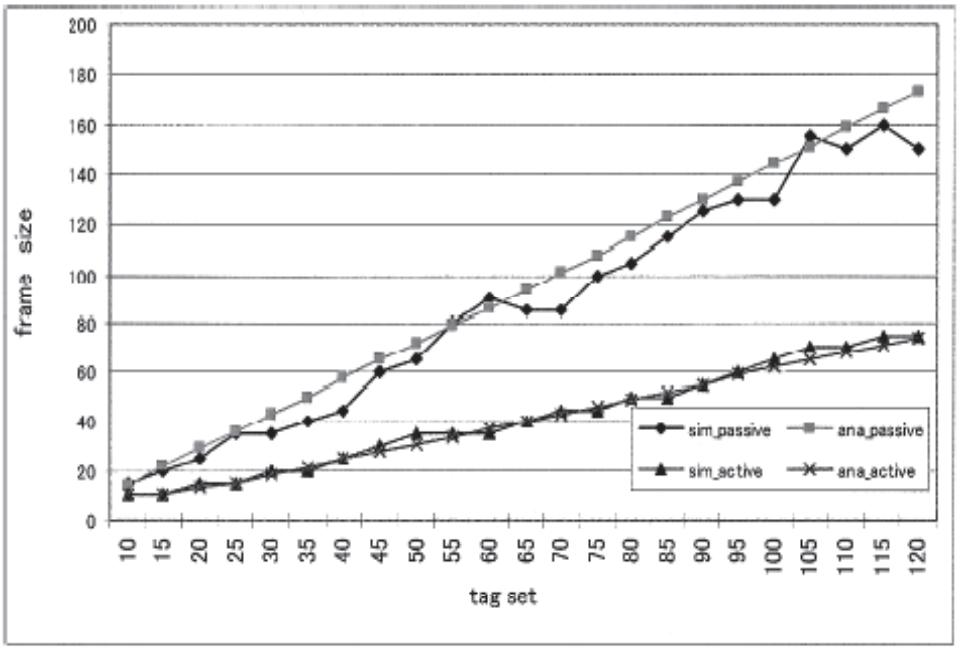


Fig. 8. Example of Optimal Frame Size For Minimum Census Delay [Bin05]

Chebyshev's inequality measures the difference the real values and expected values to estimate the number of tags for which the difference becomes minimal [Vogt02]. The number of tags are estimated using the currently used frame size (N) and the results of previous read cycle $<c_0, c_1, c_k>$ representing the number of empty slots, the number of slots filled with one tag and the number of collided slots respectively. And $< a_0^{N,n}, a_1^{N,n}, a_{\geq 2}^{N,n} >$ denotes the expected number of empty slots, the expected number of slots filled with one tag and the expected number of collided slots respectively where N and n represent the frame size and the number of tag respectively. Lower bound is more accurate for low values of n while Chebyshev's inequality is steadier for a wider range of n [Vogt02].

$$\varepsilon_{vd}(N, c_0, c_1, c_k) = \min \left| \begin{pmatrix} a_0^{N,n} \\ a_1^{N,n} \\ a_{\geq 2}^{N,n} \end{pmatrix} - \begin{pmatrix} c_0 \\ c_1 \\ c_k \end{pmatrix} \right| \quad (2)$$

```
* Function: VogtAlgorithm ()
- Variable: integer : N, N0, n_est, stepN, c, t
- N = minimum frame size
- repeat begin while stepN is smaller than maxStep:
      stepN++;
      Perform a read cycle with N and store the result in c
      Estimate number of tags with N and c, and then store result in t
      If t is bigger than n_est
      then save t in n_est.
            Call adaptFrameSize(N, n_est) and save value in N0.
            If N0 is bigger than N
            then reset stepN with 0. And, save N0 in N.
- repeat end
* Function: adaptFrameSize(N, n_est)
- Return type : integer
- repeat begin while n_est is smaller than low value for choice of N:
      Store N/2 in N.
- repeat end
- repeat begin while n_est is bigger than high value for choice of N:
      Store 2*N in N.
- repeat end
```

Fig. 9. Pseudo Code of H. Vogt's Algorithm

Figure 10 presents the optimal frame size resulting from the execution of function adaptiveFrameSize (*N*, *n*_est) using as an input the estimated number of tags and the value of the current frame size *N*. The optimal frame size is selected for a range of tags, based on a low and high margin.

N slots	1	4	8	16	32	64	128	256
low	–	–	–	1	10	17	51	112
high				9	27	56	129	∞

Fig. 10. Choosing Optimal Frame Size [Vogt02]

2.3.2 Bin ZHEN et al.'s algorithm

This algorithm estimates the number of tags using the *posteriori* probability. The *posteriori* probability of *k* tags for an observed slot is as below:

$$p_k^0(i) = \begin{cases} 0 & \text{if } k = 0, 1 \\ & \text{if } k \geq 2 \\ \dfrac{p_k(i)}{1 - p_0(i) - p_1(i)} \end{cases} \tag{3}$$

The a *posteriori* expected value of the number of tags is respectively, 0 for an empty slot, 1 for a slot filled with one tag, and $\sum_{k=2}^{N} k p_k^0(i)$ tags for a collided slot. Thus, the estimated tag sets from the current read cycle is $p_1(i) + \sum_{k=2}^{N} k p_k^0(i)$. Then, the estimated number of tags comes from the result of the *i*th read cycle is,

$$n_{est}(i+1) = s + Kg \tag{4}$$

Where K = 2.39 is a constant for collided slots [Bin05]. And, the frame size (N) in the (i+1)th read cycle is

$$\begin{aligned} N(\mathrm{i}+1) &= H * n_{est}(i+1) \\ &\begin{cases} H = 1 - 1.4 \text{ Passive tag} \\ H = 0.8 - 1 \text{ Active tag} \end{cases} \end{aligned} \tag{5}$$

where H is a constant, which maps the tags to the frame size. The update of the frame size occurs when $n_{est}(i+1) \geq \gamma * n_{est}(i)$. Here, γ is constant value which is 1.15 and it denotes a threshold to handle random jitter of number of estimated tags. The procedure of this algorithm is shown in Figure 11.

```
* Function: BinZHENAlgorithm ()
-Variable: integer : N, i, c, s, n_est(i)
 N = minimum frame size
 i = 0
- repeat begin while i is smaller than the number of read cycle for confidence level:
      i ++;
      Perform a read cycle with N and store the result in c for collided slots, s for success
      slots.
      Estimate number of tags with N, c and s, and then store result in n_est(i).
      If n_est(i) is bigger than γ * n_est(i - 1)
      then Call calculateFrameSize(n_est(i), tagType) and save value in N.
           i = 0.
- repeat end
* Function: calculateFrameSize(n_est(i), tagType)
- Return type : integer
- If tagType is passive
  then N(i) = (random value of 1 – 1.4)  * n_est(i)
  else N(i) = (random value of 0.8 – 1)  * n_est(i)
- return N(i)
```

Fig. 11. Pseudo Code of Bin ZHEN et al.'s Algorithm

2.3.3 EDFSA (Enhanced Dynamic Framed Slotted ALOHA)

This algorithm estimates the number of unread tags instead of number of tags to determine the frame size. H. Vogt's algorithm shows poor performance when the number of tags becomes large because the variance of the tag number estimation is increased according to the number of tags increase [Rom90]. Therefore, to handle the poor performance of large number of tag identification EDFSA algorithm restricts the number of responding tags as much as the frame size. Conversely, if the number of tags is too small as compared with the frame size it reduces the frame size. To estimate the number of unread tags equation (2) is used. The procedure of EDFSA algorithm's read cycle is shown in Figure 12.

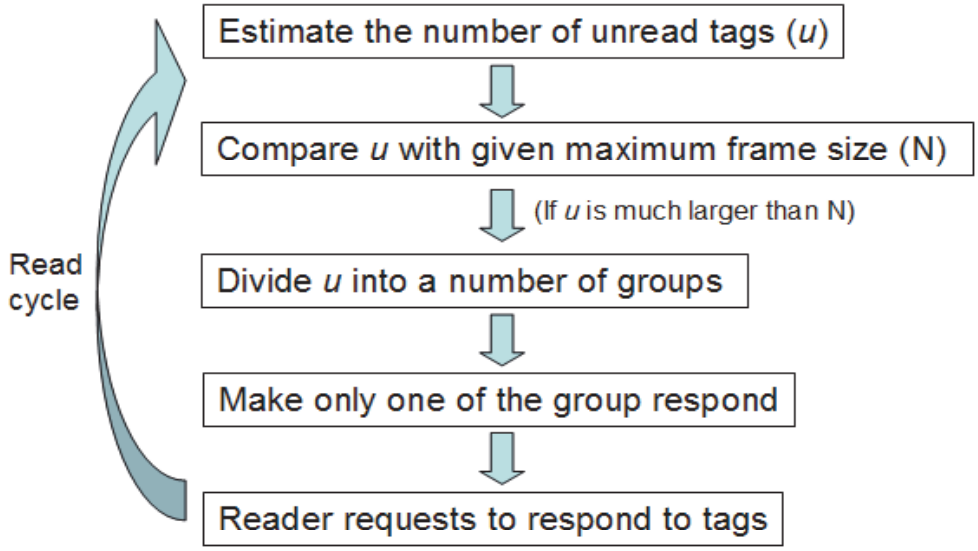


Fig. 12. Read Cycle of EDFSA Algorithm

3.1 Evaluating delays

To evaluate the implementation of the BFSA protocol I first evaluated *the total census delay* of the tag reading process. It is comprised of three different delays; *success delay, collision delay* and *idle delay*. Thus, the total census delay is defined as

$$T[n] = n + C[n] + I[n] \tag{6}$$

where n is success delay, $C[n]$ is collision delay and $I[n]$ is idle delay [Cappelletti06]. The unit of delay can be defined as a slot duration *T (sec)* and it is defined as,

$$T = \frac{\text{ID (bits)}}{\text{data\_rate (bps)}} \tag{7}$$

where *ID (bits)* is the size of the packet containing tag's ID, and *data_rate (bps)* is the data rate from tag to reader.