

## Chapter 2

# Tag Estimation in RFID Systems

### 2.1 System Model

This section introduces the tag estimation problem and the the energy issue in this problem. The communication model between RFID readers and tags is explained.

#### 2.1.1 Tag Estimation Problem

A tag estimation problem is the problem to design efficient algorithms to estimate the number of RFID tags in a deployment area without actually reading the ID of each tag. Let  $N$  be the actual number of tags and  $\hat{N}$  be the estimate. The estimation accuracy is specified by a confidence interval with two parameters: a probability value  $\alpha$  and an error bound  $\beta$ , both in the range of  $(0, 1)$ . The requirement is that the probability for  $N/\hat{N}$  to fall in the interval  $[1 - \beta, 1 + \beta]$  should be at least  $\alpha$ , i.e.,

$$Prob\{(1 - \beta)\hat{N} \leq N \leq (1 + \beta)\hat{N}\} \geq \alpha.$$

Our goal is to reduce the energy overhead incurred to the tags during the estimation process that achieves the above accuracy.

#### 2.1.2 Energy Issue

We consider RFID systems using active tags. Tagged goods (such as apparel) may stack in piles, and there may be obstacles, such as racks filled with merchandize, between a tag and the reader. We expect active tags are designed to transmit with significant power that is high enough to ensure reliable information delivery in such a demanding environment. Hence, energy cost due to the tags' transmissions is the main concern in our algorithm design; it increases at least in the square of the

maximum distance to be covered by the RFID system. Energy consumption that powers a tag's circuit for computing and receiving information is not affected by long distance and obstacles. We consider RFID systems where power consumption by tags is dominated by transmission events due to long distances that the systems need to cover. Energy consumed by the RFID reader is less of a concern. We assume the reader transmits at sufficiently high power.

### 2.1.3 Communication Protocol

The following communication protocol is used between a reader and tags. The reader first synchronizes the clocks of the tags and then performs a sequence of pollings. Clock synchronization only needs to happen at the beginning of the protocol execution. RFID systems operate in low-rate wireless channels. If an operation takes a short period of time, clock drift should not be a major issue in a low-rate channel.

In each polling, the reader sends out a request, which is followed by a slotted time frame during which the tags respond. The polling request from the reader carries a *contention probability*  $0 < p \leq 1$  and a frame size  $f$ . Each tag will participate in the current polling with probability  $p$ . If it decides to participate, it will pick a slot uniformly at random from the frame, and transmit a bit string (called *response*) in that slot. The format of the response depends on the application. If the tag decides to not participate, it will keep silent. In our solutions,  $p$  will be set in the order of  $1/N$ .

If we know a lower bound  $N_{min}$  of  $N$ , the contention probability can be implemented efficiently to conserve energy. For example, a company's inventory of certain goods may be in the thousands and never before reduced below a certain number, or the company has a policy on the minimum inventory, or the RFID estimation becomes unnecessary when the number of tags is below a threshold. In these cases, we will have a lower bound  $N_{min}$ , which can be much smaller than  $N$ . If we know such a value of  $N_{min}$ , we can implement a contention probability  $p$  without requiring all tags to participate in the contention process. Since only a small number of tags actually participate in contention, energy cost is reduced. The implementation is described as follows: At the beginning of a polling, each tag makes a probabilistic decision: It goes to a standby mode for the current polling with probability  $1 - 1/N_{min}$  and wakes up until the next polling starts, or it stays awake to receive the polling request with probability  $1/N_{min}$  and then decides to respond with probability  $\min\{p \times N_{min}, 1\}$ . For example, if  $N = 10,000$  and  $N_{min} = 1,000$ , then only 10 tags stay awake in each polling.

In the above communication protocol, the reader's request may include an optional prefix and only tags that satisfy the prefix will participate in the polling. For example, suppose all tags deployed in one section of a warehouse carry the 96-bit GEN2 IDs that begin with "000" in the Serial Number field. In order to estimate the number of tags in this section, the request carries a predicate testing whether the first three bits of a tag's Serial Number is "000".

### 2.1.4 Empty/Singleton/Collision Slots

A slot is said to be *empty* if no tag responds (transmits) in the slot. It is called a *singleton slot* if exactly one tag responds. It is a *collision slot* if more than one tag responds. A singleton or collision slot is also called a *non-empty slot*. The Philips I-Code system [12] requires a slot length of 10 bits in order to distinguish singleton slots from collision slots. On the contrary, one bit is enough if we only need to distinguish empty slots from non-empty slots — ‘0’ means empty and ‘1’ means non-empty. Hence, the response will be much shorter (or consume much less energy) if an algorithm only needs to know empty/non-empty slots, instead of all three types of slots as required by [7].

In order to prolong the lifetime of tags, there are two ways to reduce their energy consumption: reducing the size of each response and reducing the number of responses. We will present algorithms that require only the knowledge of empty/non-empty slots and employ statistical methods to minimize the amount of transmission needed from the tags.

## 2.2 Generalized Maximum Likelihood Estimation Algorithm

The first estimator for the number of RFID tags is called the *generalized maximum likelihood estimation* (GMLE) algorithm. It fully utilizes the information from all pollings in order to minimize the number of pollings it needs to meet the accuracy requirement.

### 2.2.1 Overview

GMLE uses the polling protocol described in Sect. 2.1.3. The frame size  $f$  is fixed to be one slot. The RFID reader adjusts the contention probability for each polling. Let  $p_i$  be the contention probability of the  $i$ th polling. GMLE only records whether the sole slot in each polling is empty or non-empty. Based on this information, it refines the estimate  $\hat{N}$  until the accuracy requirement is met. Let  $z_i$  be the slot state of the  $i$ th polling. When at least one tag responds, the slot is non-empty and  $z_i = 1$ . When no tag responds, it is empty and  $z_i = 0$ . The sequence of  $z_i, i \geq 1$ , forms the *response vector*.

At the  $i$ th polling, each tag has a probability  $p_i$  to transmit and, if any tag transmits,  $z_i$  will be one. Hence,

$$\text{Prob}\{z_i = 1\} = 1 - (1 - p_i)^N \approx 1 - e^{-Np_i}, \quad (2.1)$$

where  $N$  is the the actual number of tags.

If the contention probabilities of the pollings are picked too small, the response vector will contain mostly zeros. If the contention probabilities are picked too large, the response vector will contain mostly ones. Both cases do not provide sufficient statistical information for accurate estimation. As will be discussed shortly, our analysis shows that the optimal contention probability for minimizing the number of pollings is  $p_i = 1.594/N$ . The problem is that we do not know  $N$  (which is the quantity we want to estimate).

In order to determine  $p_i$ , GMLE consists of an *initialization phase* and an *iterative phase*. The former quickly produces a coarse estimation of  $N$ . The latter refines the contention probability and generates the estimation result.

### 2.2.2 Initialization Phase

We want to pick a small value for the initial contention probability  $p_1$  at the first polling. The expected number of responding tags is  $Np_1$ . If  $p_1$  is picked too large, a lot of tags will respond, which is wasteful because one response or many responses produce the same information — a non-empty slot. Suppose we know an upper bound  $N_{max}$  of  $N$ . This information is often available in practice. For example, we know  $N_{max}$  is 10,000 if the warehouse is designed to hold no more than 10,000 microwaves (each tagged with a RFID), or the company's inventory policy requires that in-store microwaves should not exceed 10,000, or the warehouse only has 10,000 RFID tags in use.  $N_{max}$  can be much bigger than  $N$ . We pick  $p_1 = 1/N_{max}$  such that the expected number of responding tags is no more than one. If  $z_1 = 0$ , we multiply the contention probability by a constant  $C(> 1)$ , i.e.,  $p_2 = p_1 \times C$  for the second polling. We continue multiplying the contention probability by  $C$  after each polling until a non-empty slot is observed. When that happens (say, at the  $l$ th polling), we have a coarse estimation of  $N$  to be  $1/p_l$ . Then we move to the next phase. When  $C$  is relatively large, the initialization phase only takes a few pollings to complete due to the exponential increase of the contention probability.

### 2.2.3 Iterative Phase

This phase iteratively refines the estimation result after each polling, and terminates when the specified accuracy requirement is met. Let  $\hat{N}_i$  be the estimated number of tags after the  $i$ th polling. To compute  $\hat{N}_i$ , the reader performs three tasks at the  $i$ th polling. First, it sets the contention probability as follows before sending out the polling request:

$$p_i = \frac{\omega}{\hat{N}_{i-1}}, \quad (2.2)$$

where  $\hat{N}_{i-1}$  is the estimate after the previous polling and  $\omega$  is a system parameter, which will be extensively analyzed in the next subsection. Second, based on the received  $z_i$  and the history information, the reader finds the new estimate of  $N$  that maximizes the following likelihood function:

$$L_i = \prod_{j=1}^i (1 - p_j)^{N(1-z_j)} (1 - (1 - p_j)^N)^{z_j}, \quad (2.3)$$

where  $(1 - p_j)^{N(1-z_j)} (1 - (1 - p_j)^N)^{z_j}$  is the probability for the observed state  $z_j$  of the  $j$ th polling to occur. Namely, we want to find

$$\hat{N}_i = \arg \max_N \{L_i\}. \quad (2.4)$$

Third, after computing  $\hat{N}_i$ , the reader has to determine if the confidence interval of the new estimate meets the requirement. In the following, we show how the above tasks can be achieved.

### 2.2.3.1 Compute the value of $\hat{N}_i$

We compute the new estimate of  $N$  that maximizes (2.3). Since the maxima is not affected by monotone transformations, we use logarithm to turn the right side of the equation from product to summation:

$$\ln(L_i) = \sum_{j=1}^i \left[ N(1 - z_j) \ln(1 - p_j) + z_j \ln(1 - (1 - p_j)^N) \right].$$

To find the maxima, we differentiate both sides:

$$\frac{\partial \ln(L_i)}{\partial N} = \sum_{j=1}^i \left[ (1 - z_j) \ln(1 - p_j) - z_j \frac{(1 - p_j)^N \ln(1 - p_j)}{1 - (1 - p_j)^N} \right]. \quad (2.5)$$

We then set the right hand side to zero and solve the equation for the new estimate  $\hat{N}_i$ . Note that the derivative is a monotone function of  $N$ , we can numerically obtain  $\hat{N}_i$  through bisection search.

### 2.2.3.2 Termination Condition

Using the  $\delta$ -method [2], we show that, when  $i$  is large,  $\hat{N}_i$  approximately follows the Gaussian distribution:

$$Norm\left(N, \frac{(1 - (1 - p_i)^N)}{i(1 - p_i)^N \ln^2(1 - p_i)}\right).$$

The variance of  $\hat{N}_i$  is

$$\text{Var}(\hat{N}_i) \approx \frac{1 - (1 - p_i)^N}{i(1 - p_i)^N \ln^2(1 - p_i)}. \quad (2.6)$$

When  $N$  is large and  $p_i$  is small, we can approximate  $(1 - p_i)^N$  as  $e^{-Np_i}$  and  $\ln(1 - p_i)$  as  $p_i$ . The above variance becomes

$$\text{Var}(\hat{N}_i) \approx \frac{e^{Np_i} - 1}{ip_i^2}. \quad (2.7)$$

Hence, the confidence interval of  $N$  is

$$\hat{N}_i \pm Z_\alpha \cdot \sqrt{\frac{e^{\hat{N}_i p_i} - 1}{ip_i^2}}, \quad (2.8)$$

where  $Z_\alpha$  is the  $\alpha$  percentile for the standard Gaussian distribution. For example, when  $\alpha = 95\%$ ,  $Z_\alpha = 1.96$ . Because  $N$  is undetermined, we use  $\hat{N}_i$  as an approximation when computing the standard deviation in (2.8).

The termination condition for GMLE is therefore

$$Z_\alpha \cdot \sqrt{\frac{e^{\hat{N}_i p_i} - 1}{ip_i^2}} \leq \hat{N}_i \cdot \beta, \quad (2.9)$$

where  $\beta$  is the error bound. The above inequality can be rewritten as

$$\sqrt{i} \geq \frac{Z_\alpha \sqrt{e^{\hat{N}_i p_i} - 1}}{\hat{N}_i p_i \beta}. \quad (2.10)$$

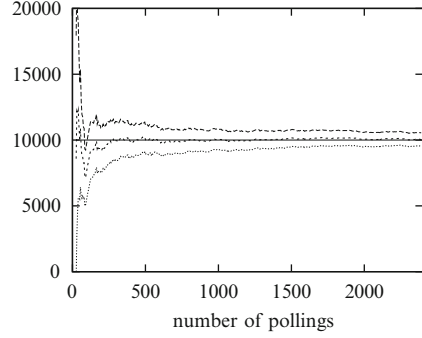
When  $i$  is large, the estimation changes little from one polling to the next. Hence,  $p_i = \omega / \hat{N}_{i-1} \approx \omega / \hat{N}_i$ . We have

$$i \geq \frac{Z_\alpha^2 \cdot (e^\omega - 1)}{\omega^2 \beta^2}. \quad (2.11)$$

Hence, if  $\omega$  is determined, we can theoretically compute the approximate number of pollings that are required in order to meet the accuracy requirement. For example, if  $\alpha = 95\%$ ,  $\beta = 5\%$ , and  $\omega = 1.594$  (which is the optimal value to be given shortly), 2372 pollings will be required. Note that (2.11) is independent with the actual number of tags,  $N$ . Hence, our approach has perfect scalability.

Figure 2.1 shows the simulation result of GMLE when  $N=10,000$ ,  $\alpha=95\%$ ,  $\beta = 5\%$  and  $\omega = 1.594$ . The simulation setup can be found in Sect. 2.4. The middle curve is the estimated number of tags,  $\hat{N}_i$ , with respect to the number pollings. It

**Fig. 2.1** The middle curve shows the estimated number of tags with respect to the number of pollings. The upper and lower curves show the confidence interval.



converges to the true value  $N$  represented by the central straight line. The upper and lower curves represent the 95% confidence interval, which shrinks as the number of pollings increases.

### 2.2.4 Determine the Value of $\omega$

We demonstrate the impact of the value  $\omega$  on two performance metrics: the *number of pollings* and the *number of tag responses* (i.e., the number of tag transmissions). The former measures the estimation time since each polling takes an equal amount of time for request/response exchange. The latter measures the energy cost because each response corresponds to one tag making one transmission in a slot.

#### 2.2.4.1 Number of Pollings

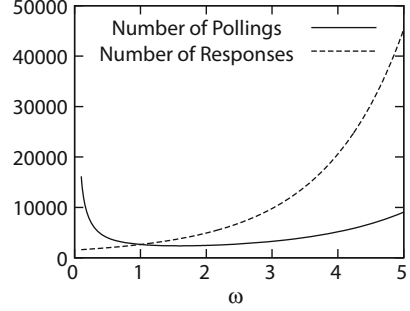
According to (2.11), the number of pollings for meeting the accuracy requirement is  $Z_\alpha^2(e^\omega - 1)/(\omega^2\beta^2)$ . To find its minimum value, we differentiate it with respect to  $\omega$  and let the result be zero. Solving the equation, we have  $\omega = 1.594$ . Hence, the optimal value of  $p_i$  that minimizes the number of pollings is

$$p_i = \frac{1.594}{\hat{N}_{i-1}}. \quad (2.12)$$

#### 2.2.4.2 Number of Responses

We count the total number of responses during the estimation process. After a small number of pollings, the estimation will closely approximate  $N$  (see Fig. 2.1). Hence, the expected number of responses for each polling is  $Np_i \approx N_{i-1}p_i = \omega$ . After  $Z_\alpha^2(e^\omega - 1)/(\omega^2\beta^2)$  pollings are made, the total number of responses is roughly

**Fig. 2.2** The solid line shows the number of pollings with respect to  $\omega$  when  $\alpha = 95\%$  and  $\beta = 5\%$ . The dotted line shows the number of responses.



$$\frac{Z_{\alpha}^2 \cdot (e^{\omega} - 1)}{\omega^2 \beta^2} \omega = \frac{Z_{\alpha}^2 \cdot (e^{\omega} - 1)}{\omega \beta^2}. \quad (2.13)$$

Simulation results will demonstrate that the approximation in the above count is reasonably accurate. It is an increasing function with respect to  $\omega$ , which means that a larger value of  $\omega$  will lead to a larger number of responses. We give the intuition as follows: A larger  $\omega$  means a larger contention probability and thus more collisions. Two or more responses in a collision slot produce the same amount of information as one response in a singleton slot (see further explanation in Sect. 2.2.6). In other words, in order to generate the necessary amount of information for meeting the accuracy requirement, more responses must be needed if there are more collisions.

### 2.2.4.3 Numerical Results

in Fig. 2.2, we plot the number of pollings and the number of responses with respect to the value of  $\omega$ . The number of pollings is minimized at  $\omega = 1.594$ . When  $\omega$  is smaller than 1.594, its value controls the performance tradeoff between the two metrics. When we decrease  $\omega$ , the energy cost (i.e., the number of responses) drops at the expenses of the estimation time (i.e., the number of pollings).

### 2.2.5 Request-less Pollings

We observe that, after a number of pollings, the value of  $p_i$  will stay in a very small range and does not change much. It becomes unnecessary for the RFID reader to transmit it at each polling. Hence, we can improve GMLE as follows: If the percentage change in  $p_i$  during a certain number  $M_1$  of consecutive pollings is below a small threshold, the reader will broadcast a polling request, carrying the latest value of  $p_i$ , a flag indicating that it will no longer transmit polling requests



for a certain number  $M_2$  of slots, and the value of  $M_2$ . Without receiving further polling requests, the tags will respond with the same contention probability in the subsequent  $M_2$  slots. This is called the *request-less pollings*. After  $M_2$  slots, the reader will recalculate the contention probability, broadcast another polling request, carrying the new probability value, a flag, and  $M_2$ . This process repeats until the termination condition in (2.9) is met. With the threshold being 10%,  $M_1 = 10$ , and  $M_2 = 50$ , simulation results show that the performance difference caused by request-less pollings is negligibly small even though the contention probability during request-less pollings may be slightly off the value set by (2.2). Request-less pollings can also be applied to the algorithm in the next section.

### 2.2.6 Information Loss due to Collision

GMLE has a frame size of one slot. It obtains only binary information at each polling. No matter how many tags respond, the information that the reader receives is always the same, i.e.,  $z_i = 1$ , which implies information loss when two or more tags decide to transmit at a polling. Let's compare two scenarios. In one scenario, only one tag responds at a polling. In the other, two tags respond. These two scenarios generate the same information but the energy cost of the second scenario is twice of the first. To address this issue, we present another algorithm that reduces the probability of collision and, moreover, compensate the impact of collision in its computation.

## 2.3 Enhanced Generalized Maximum Likelihood Estimation Algorithm

The *enhanced generalized maximum likelihood estimation* (EGMLE) algorithm also utilizes history information from previous pollings and uses the maximum likelihood method to estimate the number of tags. However, instead of only obtaining binary information, it computes the number of responses in each polling. Because more information can be extracted, it is able to achieve much better energy efficiency than GMLE.

### 2.3.1 Overview

EGMLE uses the same polling protocol as GMLE does, except that its frame size  $f$  is larger than one in order to reduce the probability of collision. The result of the  $i$ th polling,  $x_i$ , is no longer a binary value. Instead, it is an estimate of the number of tags that respond during the polling.

EGMLE takes two steps to solve the collision problem. First, it increases the frame size  $f$  such that the tags that decide to respond at a polling are likely to respond at different slots in the frame. We pick values for  $p_i$  and  $f$  such that the collision probability is very small. Second, we compensate the remaining impact of collision in our computation.

EGMLE also consists of an *initialization phase* and an *iterative phase*. The initialization phase of EGMLE is the same as the initialization phase of GMLE, except that when the RFID reader obtains the first non-zero result  $x_l$  at the  $l$ th polling with a contention probability  $p_l$ , it computes a coarse estimation of  $N$  as  $x_l/p_l$ . Then it moves to the next phase below.

### 2.3.2 Iterative Phase

This phase iteratively refines the estimation after each polling, and terminates when the specified accuracy requirement is met. The reader performs four tasks during the  $i$ th polling. First, it computes the contention probability before sending out the polling request.

$$p_i = \frac{\omega}{\hat{N}_{i-1}}, \quad (2.14)$$

where  $\hat{N}_{i-1}$  is the estimate after the previous polling and  $\omega$  is one by default. As we will show in the next subsection, performance tradeoff can be made by choosing other values for  $\omega$ .

Second, the reader computes the number of responses  $x_i$  in the current frame.

Third, based on the received  $x_i$  and the history information, the reader computes the new estimate of  $N$  that maximizes the following likelihood function:

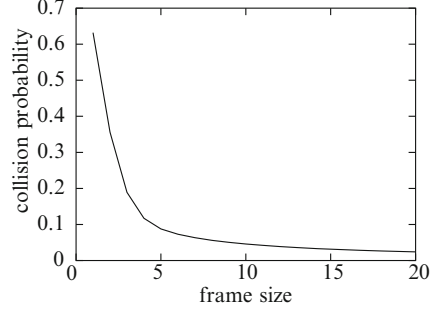
$$L_i = \prod_{j=l+1}^i \left[ \frac{1}{\sqrt{2\pi N p_j (1 - p_j)}} \cdot e^{-\frac{((1+\varepsilon)x_j - N p_j)^2}{2N p_j (1 - p_j)}} \right], \quad (2.15)$$

where  $\varepsilon$  is introduced to compensate for collision and the iterative phase begins from the  $(l+1)$ th polling. The above formula and the value of  $\varepsilon$  will be derived shortly. The new estimate is

$$\hat{N}_i = \arg \max_N \{L_i\}. \quad (2.16)$$

Fourth, after computing  $\hat{N}_i$ , the reader determines if the estimate meets the accuracy requirement. In the following, we give the details of the above tasks.

**Fig. 2.3** The collision probability with respect to the frame size  $f$ .



### 2.3.2.1 Compute the number of responses

At the  $i$ th polling, the reader measures the number of non-empty slots in the frame, denoted as  $x_i$ , which is an integer in the range of  $[0..f]$ . Due to possible collision, the actual number of responses, denoted as  $x_i^*$ , can be greater. Let  $x_i^* = (1 + \varepsilon)x_i$ . The value of  $\varepsilon$  is determined below.

Since each tag independently decides to respond with probability  $p_i$ ,  $x_i^*$  follows a binomial distribution,  $Bino(N, p_i)$ , i.e.,

$$Prob\{x_i^* = k\} = \binom{N}{k} p_i^k (1 - p_i)^{N-k}. \quad (2.17)$$

Suppose  $\omega$  takes the default value, 1. When  $i$  is large,  $N_{i-1}$  approximates  $N$  and thus  $p_i \approx 1/N$ . If  $N$  is sufficiently large,  $Prob\{x_i^* = 2\} \approx 0.1839$ ,  $Prob\{x_i^* = 3\} \approx 0.0613$ ,  $Prob\{x_i^* = 4\} \approx 0.0153$ , and the probability decreases exponentially with respect to  $k$ .  $Prob\{x_i^* > 4\}$  is only about 0.0037.

Next, we compute the probability for collision to happen at the  $i$ th polling, which is denoted as  $Prob_i\{collision\}$ .

$$\begin{aligned} Prob_i\{collision\} &= \sum_{k=2}^N Prob_i\{collision|x_i^* = k\} \times Prob\{x_i^* = k\} \\ &= \sum_{k=2}^f \left(1 - \frac{P(f, k)}{f^k}\right) \times Prob\{x_i^* = k\} + \sum_{k=f+1}^N 1 \times Prob\{x_i^* = k\}, \end{aligned}$$

where  $P(f, k) = f!/(f-k)!$  is the permutation function.

Figure 2.3 shows the collision probability  $Prob_i\{collision\}$  with respect to  $f$ . It diminishes quickly as  $f$  increases. When  $f = 10$  (which is what we use in the simulations),  $Prob_i\{collision\}$  is just 0.046. With such a small probability, the chance for more than two tags involved in a collision or more than one collision at a polling is exceedingly small and thus ignored. Therefore, to approximate  $x_i^*$ , we multiply  $x_i$  by 1.046 to compensate the impact of collision. Namely,  $\varepsilon = 0.046$ .

### 2.3.2.2 Compute the value of $\hat{N}_i$

Recall that the iterative phase starts at the  $(l+1)$ th polling. After the  $i$ th polling, the reader has collected the values of  $x_j$ ,  $l < j \leq i$ . By our previous analysis, we know that  $x_j^* = (1 + \varepsilon)x_j$  and it follows a binomial distribution  $Bino(N, p_j)$ . When  $N$  is large enough, the binomial distribution can be closely approximated by a Gaussian distribution  $Norm(\mu_j, \sigma_j)$  with parameters  $\mu_j = Np_j$  and  $\sigma_j = \sqrt{Np_j(1-p_j)}$ . Namely,

$$x_j^* \approx (1 + \varepsilon)x_j \sim Norm(Np_j, Np_j(1-p_j)). \quad (2.18)$$

Hence, the probability for the *measured number of responses*,  $(1 + \varepsilon)x_j$ , to occur under this distribution is  $[2\pi Np_j(1-p_j)]^{-1/2} \exp\{-[(1 + \varepsilon)x_j - Np_j]^2 / [2Np_j(1-p_j)]\}$ . The likelihood function for all measured numbers of responses in the pollings,  $(1 + \varepsilon)x_j$ ,  $l < j \leq i$ , to occur is

$$L_i = \prod_{j=l+1}^i \left[ \frac{1}{\sqrt{2\pi Np_j(1-p_j)}} \cdot e^{-\frac{((1+\varepsilon)x_j - Np_j)^2}{2Np_j(1-p_j)}} \right]. \quad (2.19)$$

To find the value  $\hat{N}_i$  that maximizes the likelihood function, we first take logarithm on both sides of (2.19),

$$\ln(L_i) = \sum_{j=l+1}^i \left[ \ln \frac{1}{\sqrt{2\pi Np_j(1-p_j)}} - \frac{((1 + \varepsilon)x_j - Np_j)^2}{2Np_j(1-p_j)} \right]. \quad (2.20)$$

We then differentiate both sides,

$$\begin{aligned} \frac{\partial \ln(L_i)}{\partial N} &= \sum_{j=l+1}^i \left[ -\frac{1}{2N} + \frac{(1 + \varepsilon)^2 x_j^2 - (Np_j)^2}{2N^2 p_j(1-p_j)} \right] \\ &= \sum_{j=l+1}^i \frac{(1 + \varepsilon)^2 x_j^2 - (Np_j)^2}{2N^2 p_j(1-p_j)} - \frac{i-l}{2N}. \end{aligned} \quad (2.21)$$

Finally, we set the right side to be zero and numerically compute the value of  $\hat{N}_i$ .

### 2.3.2.3 Termination Condition

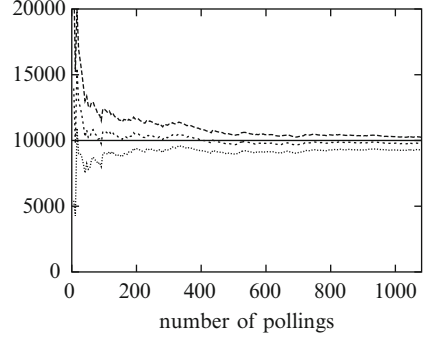
The *fisher information*<sup>1</sup>  $\mathcal{J}(\hat{N}_i)$  of  $L_i$  is defined as follows

$$\mathcal{J}(\hat{N}_i) = -E \left[ \frac{\partial^2 \ln(L_i)}{\partial N^2} \right]. \quad (2.22)$$

---

<sup>1</sup>The fisher information [9] is a way of measuring the amount of information that an observable random variable  $x$  carries about an unknown parameter  $\theta$  upon which the likelihood function of  $\theta$ ,  $L(\theta) = f(x; \theta)$ , depends.

**Fig. 2.4** The middle curve shows the estimated number of tags with respect to the number of pollings. The upper and lower curves show the confidence interval.



According to (2.21), we have

$$\begin{aligned} \mathcal{J}(\hat{N}_i) &= E \left[ \sum_{j=l+1}^i \frac{(1+\varepsilon)^2 x_j^2}{N^3 p_j (1-p_j)} - \frac{i-l}{2N^2} \right] \\ &= \sum_{j=l+1}^i \frac{(Np_j)^2 + Np_j(1-p_j)}{N^3 p_j (1-p_j)} - \frac{i-l}{2N^2} \end{aligned} \quad (2.23)$$

$$= \sum_{j=l+1}^i \frac{p_j}{N(1-p_j)} + \frac{i-l}{2N^2}. \quad (2.24)$$

Above, we have applied  $E((1+\varepsilon)^2 x_j^2) = (Np_j)^2 + Np_j(1-p_j)$  in (2.23) because  $(1+\varepsilon)x_j \sim \text{Norm}(Np_j, Np_j(1-p_j))$  and  $E(x^2) = (E(x))^2 + \text{Var}(x)$ .

Following the classical theory for MLE, when  $i$  is sufficiently large, the distribution of  $\hat{N}_i$  is approximated by

$$\text{Norm} \left( N, \frac{1}{\mathcal{J}(\hat{N}_i)} \right). \quad (2.25)$$

Hence, the confidence interval is

$$\hat{N}_i \pm Z_\alpha \cdot \sqrt{\frac{1}{\mathcal{J}(\hat{N}_i)}}. \quad (2.26)$$

Note that we use  $\hat{N}_i$  as an approximation for  $N$  in the computation when necessary since  $N$  is unknown. The termination condition for EGMLE to achieve the required accuracy is

$$Z_\alpha \cdot \sqrt{\frac{1}{\mathcal{J}(\hat{N}_i)}} \leq \hat{N}_i \cdot \beta. \quad (2.27)$$

Figure 2.4 shows the simulation result of EGMLE when  $N = 10,000$ ,  $\alpha = 95\%$ ,  $\beta = 5\%$ , and  $\omega = 1$ . The middle curve is the value of  $\hat{N}_i$ , which converges to the

value of  $N$  represented by the central straight line. The upper and lower curves represent the 95% confidence interval, which shrinks as the number of pollings increases. The algorithm terminates after 1081 pollings.

### 2.3.3 Performance Tradeoff

According to (2.14), the contention probability is proportional to  $\omega$ . We study how the value of  $\omega$  controls the tradeoff between the estimation time and the energy cost, which are measured by the number of pollings and the number of responses, respectively.

#### 2.3.3.1 Number of Pollings

Since the MLE approach provides statistically consistent estimate, when  $i$  is large, (2.24) can be approximated as follows:

$$\begin{aligned}\mathcal{J}(\hat{N}_i) &= \sum_{j=l+1}^i \frac{p_j}{N(1-p_j)} + \frac{i-l}{2N^2} \\ &\approx \left( \frac{p_i}{N(1-p_i)} + \frac{1}{2N^2} \right) \cdot (i-l) \\ &\approx \frac{2Np_i + 1}{2N^2} \cdot (i-l),\end{aligned}\tag{2.28}$$

where  $p_i \ll 1$ . According to (2.27), we have

$$\mathcal{J}(\hat{N}_i) \geq \left( \frac{Z_\alpha}{\hat{N}_i \cdot \beta} \right)^2\tag{2.29}$$

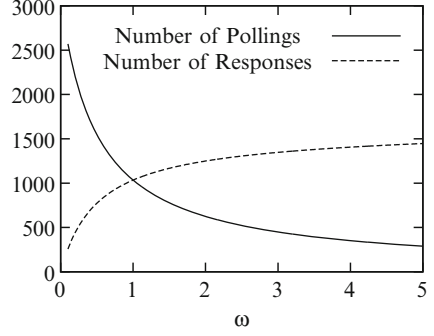
(2.28) and (2.29) give us the following inequality:

$$\begin{aligned}\frac{2Np_i + 1}{2N^2} \cdot (i-l) &\geq \left( \frac{Z_\alpha}{\hat{N}_i \cdot \beta} \right)^2, \\ i &\geq \frac{2Z_\alpha^2}{(2\omega + 1)\beta^2},\end{aligned}\tag{2.30}$$

where  $\hat{N}_i \approx N$  and  $l \ll i$ . Hence, the number of pollings it takes to achieve the accuracy requirement is  $2Z_\alpha^2/[(2\omega + 1)\beta^2]$ .

The solid line in Fig. 2.5 shows the number of pollings with respect to  $\omega$  when  $\alpha = 95\%$  and  $\beta = 5\%$ . It is a decreasing function in  $\omega$ . The reason is that a

**Fig. 2.5** The solid line shows the number of pollings with respect to  $\omega$  when  $\alpha = 95\%$  and  $\beta = 5\%$ . The dotted line shows the number of responses.



larger  $\omega$  results in more responses (and thus more information) in each polling. Consequently, a less number of pollings is needed to achieve a certain accuracy requirement.

### 2.3.3.2 Number of Responses

When  $i$  is large, the expected number of responses for each polling is  $Np_i \approx N_{i-1}p_i = \omega$ . After  $2Z_\alpha^2/[(2\omega + 1)\beta^2]$  pollings are made, the total number of responses is roughly

$$\frac{Z_\alpha^2 \cdot (e^\omega - 1)}{\omega^2 \beta^2} \omega = \frac{Z_\alpha^2 \cdot (e^\omega - 1)}{\omega \beta^2}. \quad (2.31)$$

The dotted line in Fig. 2.5 shows the number of responses with respect to  $\omega$  when  $\alpha = 95\%$  and  $\beta = 5\%$ . It is an increasing function in  $\omega$ , which means that a larger value of  $\omega$  will lead to a larger number of responses.

## 2.4 Simulations

We evaluate the performance of GMLE and EGMLE by simulations. In order to demonstrate the performance tradeoff between energy cost and estimation time, we choose two different contention probability parameters for each of the two algorithms. We use  $\omega = 0.5$  and  $1.594$  for GMLE, i.e.,  $p_i = 0.5/\hat{N}_{i-1}$  and  $1.594/\hat{N}_{i-1}$ . Note that  $1.594$  is the optimal value of  $\omega$  for time efficiency in GMLE. We denote the corresponding variants of the algorithm as GMLE(0.5) and GMLE(1.594).

For EGMLE, Fig. 2.5 shows that the number of pollings and the number of responses are both monotonic functions with respect to  $\omega$ , which means there is no optimal  $\omega$  for either energy efficiency or time efficiency. We choose  $\omega = 0.5$  and

**Table 2.1** Number of Responses when  $\alpha = 90\%$ ,  $\beta = 9\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	432S	767 S	172 S	225 S	6345 L	709 L	4342 S
10000	414S	832 S	180 S	231 S	11986 L	899 L	8683 S
20000	402S	844 S	186 S	213 S	22895 L	977 L	17366 S

1.0 for EGMLE, i.e.,  $p_i = 0.5/\hat{N}_{i-1}$  and  $1.0/\hat{N}_{i-1}$ . The corresponding variants of the algorithm is denoted as EGMLE(0.5) and EGMLE(1.0). Section 2.3.2 shows how to compute the compensation parameter  $\varepsilon$  for EGMLE(1.0), which is 0.046. Following the same steps, we obtain  $\varepsilon = 0.012$  for EGMLE(0.5). We compare the algorithms with the state-of-the-art algorithms in the related work. They are the Unified Probabilistic Estimator (UPE) [7] and the Enhanced Zero-Based (EZB) estimator [8]. The original UPE, denoted as UPE-O, is very energy-inefficient because its contention probability begins from 100% and thus all tags will respond. We modify it (denoted as UPE-M) to begin from a small initial contention probability  $1/N_{max}$  and keep the remaining part of UPE-O. This section shows the performance of both UPE-O and UPE-M. We run each simulation 100 times and average the outcomes.

In the initialization phase of our algorithms, let  $N_{max} = 1,000,000$  and  $C = 2$ . The frame size in EGMLE(0.5) and EGMLE(1.0) is 10 slots. The parameters for UPE and EZB are chosen based on the original papers whenever possible. All algorithms except for UPE need only to identify empty and non-empty slots. To set a non-empty slot apart from an empty slot, a tag only needs to respond with a short bit string (one bit) to make the channel busy. UPE has to identify empty, singleton and collision slots. To set a singleton slot apart from a collision slot, many more bits (10 used by UPE) are necessary [1]. For example, CRC may be used to detect collision.

The energy cost of an algorithm depends on (1) the number of responses that all tags transmit before the algorithm terminates and (2) the size of each response. We use ‘S’ to mean that the response is a short bit string (in the empty/non-empty case), and ‘L’ to mean a long bit string (in the empty/singleton/collision case).

We do not include the simulation results for LoF [11] because its energy cost is much higher than others. Its number of responses transmitted by the tags is  $kN$ , where  $k$  is the number of frames used in the estimation process.

### 2.4.1 Number of Responses

The first simulation studies the number of responses in each algorithm with respect to  $N$ ,  $\alpha$  and  $\beta$ . Table 2.1 shows the number of responses with respect to  $N$  when  $\alpha = 90\%$  and  $\beta = 9\%$ . GMLE and EGMLE require fewer responses than UPE and EZB. As predicted, UPE-O is energy-inefficient; UPE-M works much better.



**Table 2.2** Number of Responses when  $\alpha = 90\%$ ,  $\beta = 6\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	1041 S	1855 S	402 S	523 S	7144 L	1811 L	7236 S
10000	1153 S	1924 S	414 S	519 S	12645 L	1687 L	14472 S
20000	1015 S	1797 S	375 S	503 S	23808 L	1814 L	28944 S

**Table 2.3** Number of Responses when  $\alpha = 90\%$ ,  $\beta = 3\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	3927S	7341 S	1499 S	2037 S	12664 L	6426 L	27497 S
10000	3760S	7339 S	1489 S	2059 S	18023 L	6581 L	54993 S
20000	3783S	7350 S	1543 S	2002 S	28708 L	6993 L	109987 S

**Table 2.4** Number of Responses when  $\alpha = 95\%$ ,  $\beta = 9\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	603S	1112 S	258 S	330 S	6715 L	1073 L	4342 S
10000	669S	1120 S	247 S	304 S	12062 L	961 L	8683 S
20000	680S	1197 S	262 S	320 S	23345 L	1136 L	17366 S

**Table 2.5** Number of Responses when  $\alpha = 95\%$ ,  $\beta = 6\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	1340 S	2515 S	581 S	736 S	7712 L	2598 L	10130 S
10000	1354 S	2511 S	596 S	736 S	13477 L	2318 L	20261 S
20000	1381 S	2630 S	555 S	749 S	24631 L	2510 L	40521 S

The best algorithm is EGMLE(0.5), whose number of responses is about one fifth of what UPE-M requires and one ninetieth of what EZB requires when  $N$  is 20,000. Moreover, each response in UPE is much longer.

GMLE(0.5) has a smaller energy cost than GMLE(1.594). For example,  $N = 10,000$ , the ratio between the number of responses by GMLE(1.594) and that by GMLE(0.5) is 2.01, which is close to the theoretically-computed ratio of 1.90 in Fig. 2.2. Similarly, EGMLE(0.5) is more energy efficient than EGMLE(1.0). When  $N = 10,000$ , the ratio between the number of responses by GMLE(1.594) and that by GMLE(0.5) is 1.28, which is also close to the theoretical value of 1.34 in Fig. 2.5.

We vary  $\alpha$  from 90% to 95% and to 99%, and vary  $\beta$  from 9% to 6% and to 3%. Tables 2.2 to 2.9 show similar comparison under different values of  $\alpha$  and  $\beta$  values. In all cases, the number of responses increases when  $\alpha$  increases or  $\beta$  decreases, and except for EZB, the number does not vary much with respect to  $N$ , meaning that all algorithms except for EZB achieve good scalability. The ratio between the numbers for different algorithms appears to be quite stable under different parameter settings.

**Table 2.6** Number of Responses when  $\alpha = 95\%, \beta = 3\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	5687 S	10493 S	2181 S	2915 S	14678 L	8858 L	39074 S
10000	5673 S	10286 S	2267 S	2924 S	20845 L	9364 L	78148 S
20000	5588 S	10637 S	2217 S	2990 S	32339 L	9683 L	156297 S

**Table 2.7** Number of Responses when  $\alpha = 99\%, \beta = 9\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	1040 S	2162 S	427 S	453 S	7240 L	1726 L	7236 S
10000	1071 S	2135 S	416 S	529 S	12842 L	1906 L	14472 S
20000	1017 S	1916 S	439 S	573 S	23982 L	1819 L	28944 S

**Table 2.8** Number of Responses when  $\alpha = 99\%, \beta = 6\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	2527 S	4785 S	965 S	1269 S	9679 L	4311 L	17366 S
10000	2527 S	4637 S	973 S	1248 S	15336 L	4130 L	34733 S
20000	2440 S	4580 S	991 S	1293 S	26128 L	4044 L	69465 S

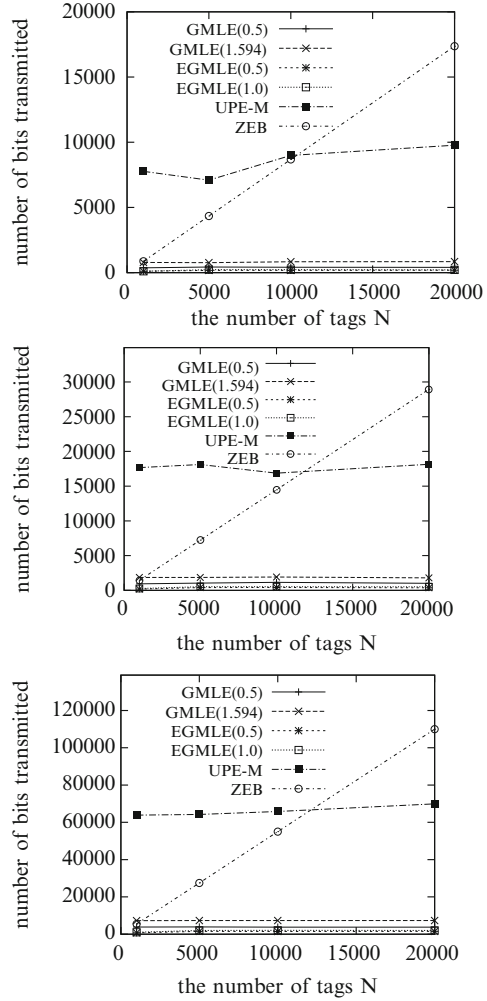
**Table 2.9** Number of Responses when  $\alpha = 99\%, \beta = 3\%$ 

N	Total number of responses						
	GMLE(0.5)	GMLE(1.594)	EGMLE(0.5)	EGMLE(1.0)	UPE-O	UPE-M	EZB
5000	9693 S	18690 S	3818 S	4993 S	21823 L	16705 L	65124 S
10000	9606 S	18223 S	3791 S	4998 S	27667 L	15882 L	130247 S
20000	9385 S	17735 S	3847 S	5027 S	38935 L	16471 L	260495 S

### 2.4.2 Total Number of Bits Transmitted

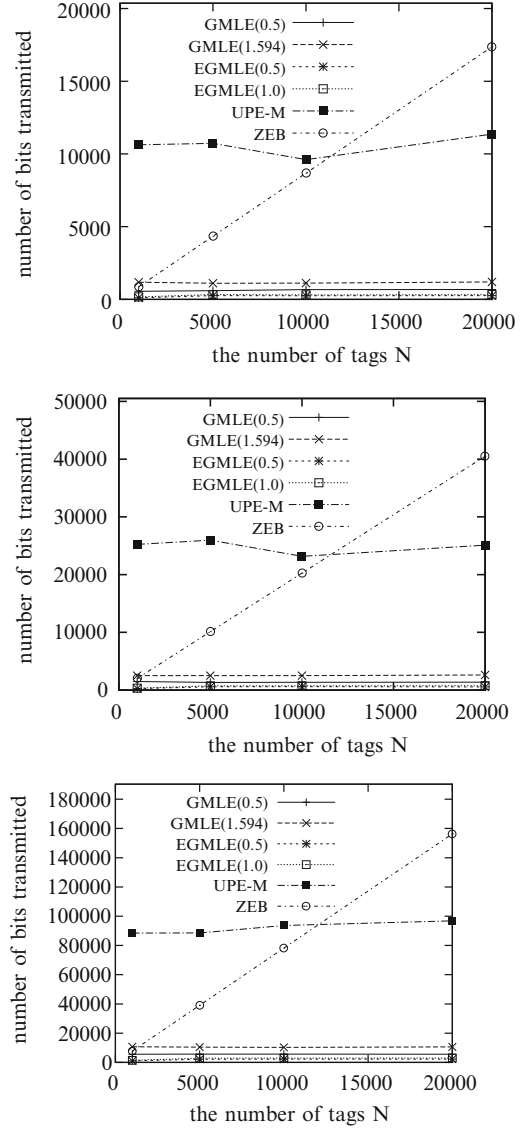
The second simulation evaluates the energy cost of the algorithms. As mentioned before, one bit is enough to separate empty/non-empty slot. Hence, the response of GMLE, EGMLE and EZB is one bit long. A response in UPE-M is 10 bits long [7]. We compare the total number of bits transmitted by all tags before each algorithm terminates. We omit the results for UPE-O, which are much worse than the results of UPE-M. Figure 2.6 shows the simulation results with respect to  $N$  when  $\alpha = 90\%, \beta = 9\%, 6\%$  and  $3\%$ . For example, when  $\alpha = 90\%, \beta = 3\%$ , and  $N = 20,000$ , the ratio between the number of bits transmitted by UPE-M (EZB) and

**Fig. 2.6** Numbers of bits transmitted when  $\alpha = 90\%$ ,  $\beta = 9\%$ ,  $6\%$  and  $3\%$ .



that by our best estimator EGMLE(0.5) is 45.32 (71.28). Figures 2.7 and 2.8 show the comparison under different  $\beta$  values when  $\alpha = 95\%$  and  $99\%$ , respectively. Their results are similar to Fig. 2.6. It should be noted that the number of bits transmitted is not an accurate measurement of the energy cost because it ignores the energy spent to power up the radio and synchronize with the reader. However, combining the number of bits and the number of transmissions (in the previous subsection) still gives a good idea on how energy-efficient each algorithm is.

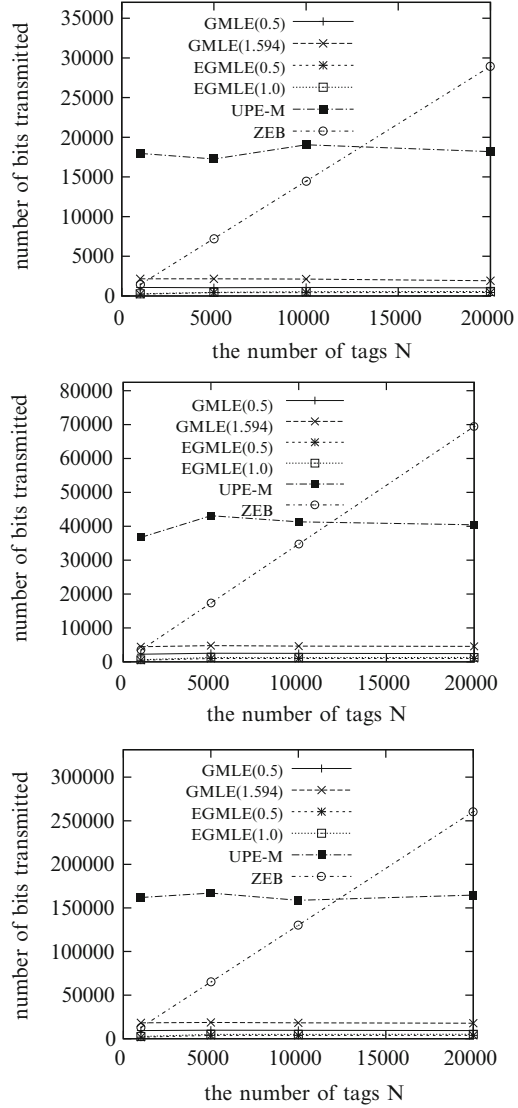
**Fig. 2.7** Numbers of bits transmitted when  $\alpha = 95\%$ ,  $\beta = 9\%$ ,  $6\%$  and  $3\%$ .



### 2.4.3 Estimation Time

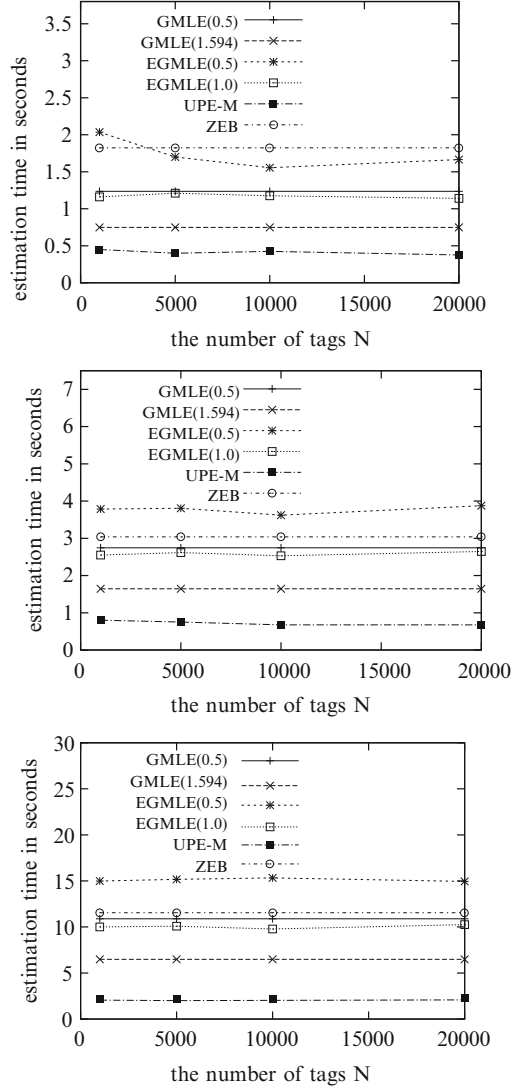
The third simulation compares the time it takes for each algorithm to complete the estimation of  $N$ . Based on the specification of the Philips I-Code system [12], after the required waiting times (e.g., gap between transmissions) are included, it can be calculated that a RFID reader needs  $0.4\text{ ms}$  to detect an empty slot,  $0.8\text{ ms}$  to detect a collision or a singleton slot, and  $1\text{ ms}$  to broadcast a polling request. Hence, GMLE, EGMLE and EZB requires a slot length of  $0.4\text{ ms}$ , while UPE-M

**Fig. 2.8** Numbers of bits transmitted when  $\alpha = 99\%$ ,  $\beta = 9\%$ ,  $6\%$  and  $3\%$ .



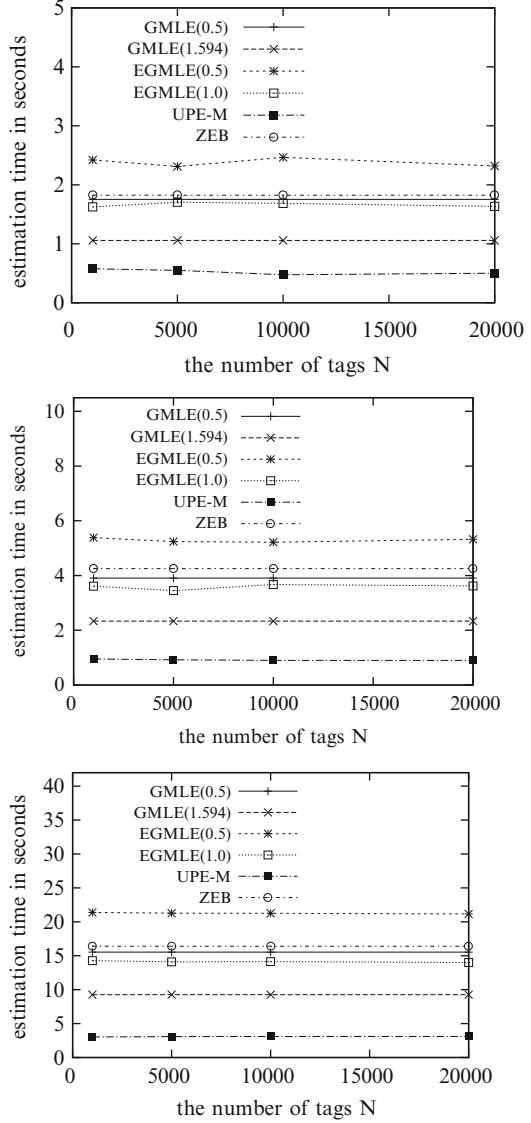
requires a slot length of  $0.8\text{ ms}$ . Recall that the contention probability takes the form of  $\omega/\hat{N}_i$ , where  $\omega$  is a known constant. Thus the reader transmits  $\hat{N}_i$  instead of the actual probability value in the polling requests. If we assume  $N_{max}$  is no more than a million, then 20 bits for  $\hat{N}_i$  are sufficient. GMLE has a fixed frame size of one slot. EGMLE has a fixed frame size of 10 slots. EZB and UPE-M also have pre-determined frame sizes. Let  $\alpha = 90\%$ ,  $\beta = 9\%$ ,  $6\%$  and  $3\%$ . The three plots in Fig. 2.9 show the estimation times of the algorithms with respect to the number of tags in the deployment. The times grow very slowly as the number of tags increase, which suggests the algorithms all scale well. In the first plot of Fig. 2.9, UPE-M

**Fig. 2.9** Estimation times of the algorithms when  $\alpha = 90\%$ ,  $\beta = 9\%$ ,  $6\%$  and  $3\%$ .



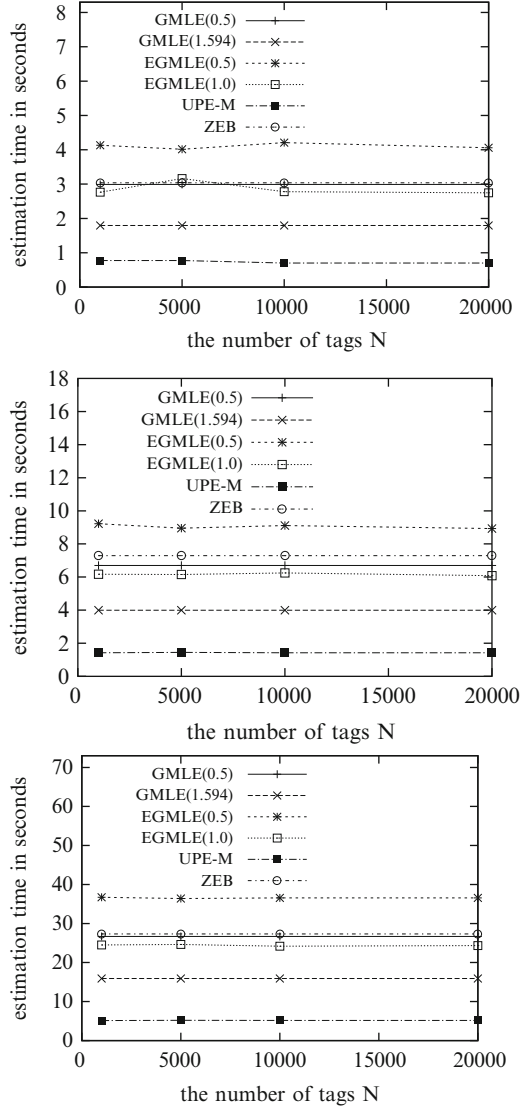
takes the least amount of time, only about 0.5 second, to estimate 20,000 tags, while the other algorithms take between 0.7 to 2.0 seconds. GMLE(1.594) takes less estimation time than GMLE(0.5) and the ratio is 0.61, which is consistent with the theoretical value of 0.58 in Fig. 2.2. Similarly, EGMLE(1.0) takes less time than EGMLE(0.5) and the ratio is 0.68, which is also consistent with the theoretical value of 0.67 in Fig. 2.5. Figures 2.10 and 2.11 show similar simulation results when  $\alpha = 95\%$  and  $99\%$ , respectively. Even though the new algorithms take longer to complete, their estimation time is still small. We believe the extra time needed can be well justified for the large energy saving.

**Fig. 2.10** Estimation times of the algorithms when  $\alpha = 95\%$ ,  $\beta = 9\%$ ,  $6\%$  and  $3\%$ .



There exists a performance tradeoff between GMLE and EGMLE. In the previous two subsections, we have examined energy cost in terms of number of responses and number of transmitted bits. EGMLE always performs better than GMLE. In this subsection, we compare estimation time of our two methods. GMLE performs better than EGMLE. Because the focus of this work is on energy efficiency, we regard EGMLE as the best estimator for energy saving.

**Fig. 2.11** Estimation times of the algorithms when  $\alpha = 99\%$ ,  $\beta = 9\%$ ,  $6\%$  and  $3\%$ .



## 2.5 Other Methods

Instead of identifying individual RFID tags, Floerkemeier [4, 5] studies the problem of estimating the cardinality of a tag set based on the number of empty slots. The proposed scheme employs a Bayesian probability estimation to achieve fast estimation. The scheme is similar to hash-based estimators [3, 14] and the difference is discussed in [8]. In Kodialam and Nandagopal's approach [7], information from tags are collected by a RFID reader in a series of time frames. Each frame consists



of a number of slots, and the tags probabilistically respond in those slots. Using the probabilistic counting methods, the reader estimates the number of tags based on the number of empty slots or the number of collision slots in each frame. Their best estimator is called the Unified Probabilistic Estimator (UPE). A follow-up work by the same authors proposes the Enhanced Zero-Based Estimator (EZB) [8], which makes its estimation based on the number of empty slots. The focus of the above estimators is to reduce the time it takes a reader to complete the estimation process. Because their goal is not conserving energy for active tags, their design is not geared towards reducing the number of transmissions made by the tags.

The Lottery-Frame scheme (LoF) [11] by Qian et al. employs a geometric distribution-based scheme to determine which slot in a time frame each tag will respond. It significantly reduces the estimation time when comparing with UPE. However, every tag must respond in each of the time frames, resulting in large energy cost when active tags use their own power to transmit. The First Non-Empty slots Based algorithm (FNEB) [6] uses the slot number of the first reply from tags in a frame to count RFID tags in both static and dynamic environments.

Also related is a novel security protocol proposed by Tan et al. to monitor the event of missing tags in the presence of dishonest RFID readers [13]. In order to prevent a dishonest reader from replaying previously collected information, they maintain a timer in the server and periodically update the system clock. Li et al. [10] design a series of efficient protocols that employ novel techniques to identify missing tags in large-scale RFID systems.

## 2.6 Summary

This chapter presents two probabilistic algorithms for estimating the number of RFID tags in a region. Solving the tag estimation problem incurs energy cost both at the RFID reader and at active tags. The asymmetry is that energy cost at tags should be minimized while energy cost at the reader is relatively less of a concern because the reader's battery can be replaced easily or it may be powered by an external source. To exploit this asymmetry, the probabilistic algorithms trade more energy cost at the reader for less cost at the tags. The performance of the algorithms is controlled by a parameter  $\omega$ , specifying the contention probability that tags use to decide whether they will transmit. By modifying this parameter, the algorithms can make tradeoff between energy cost and estimation time.

## References

1. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz Version 1.0.9. [http://www.epcglobalinc.org/standards/uhfclg2/uhfclg2\\_1.0.9-standard-20050126.pdf](http://www.epcglobalinc.org/standards/uhfclg2/uhfclg2_1.0.9-standard-20050126.pdf) (2005)
2. Casella, G., Berger, R.L.: Statistical Inference. 2nd edition, Duxbury Press (2002)

3. Durand, M., Flajolet, P.: LogLog Counting of Large Cardinalities. Proc. of European Symposium on Algorithms (2003)
4. Floerkemeier, C.: Transmission Control Scheme for Fast RFID Object Identification. IEEE Percom Workshop on Pervasive Wireless Networking (2006)
5. Floerkemeier, C., Wille, M.: Comparison of Transmission Schemes for Framed ALOHA based RFID Protocols. Workshop on RFID and Extended Network Deployment of Technologies and Applications (2006)
6. Han, H., Sheng, B., Tan, C.C., Li, Q., Mao, W., Lu, S.: Counting RFID Tags Efficiently and Anonymously. Proc. of IEEE INFOCOM (2010)
7. Kodialam, M., Nandagopal, T.: Fast and Reliable Estimation Schemes in RFID Systems. Proc. of ACM MOBICOM (2006)
8. Kodialam, M., Nandagopal, T., Lau, W.: Anonymous Tracking using RFID tags. Proc. of IEEE INFOCOM (2007)
9. Lehmann, Casella, G.: Theory of Point Estimation. Springer, 2nd edition (1998)
10. Li, T., Chen, S., Ling, Y.: Identifying the Missing Tags in a Large RFID System. Proc. of ACM Mobihoc (2010)
11. Qian, C., Ngan, H., Liu, Y.: Cardinality Estimation for Large-scale RFID Systems. Proc. of IEEE PerCom (2008)
12. Semiconductors, P.: I-CODE Smart Label RFID Tags. [http://www.nxp.com/documents/data\\_sheet/SL092030.pdf](http://www.nxp.com/documents/data_sheet/SL092030.pdf) (2004)
13. Tan, C., Sheng, B., Li, Q.: How to Monitor for Missing RFID Tags. Proc. of IEEE ICDCS (2008)
14. Whang, K., Vander-Zanden, B., Taylor, H.: A Linear Time Probabilistic Counting Algorithm for Database Applications. ACM Transactions on Database Systems (1990)

RFID as an Infrastructure

Qiao, Y.; Chen, S.-S.; Li, T.

2013, VII, 82 p. 22 illus., Softcover

ISBN: 978-1-4614-5229-4