Energy-efficient Polling Protocols in RFID Systems

Yan Qiao, Shigang Chen, Tao Li
Computer & Information Science & Engineering
University of Florida, Gainesville, FL, USA
{yqiao, sgchen,tali}@cise.ufl.edu

Shiping Chen
Network Information Center
University of Shanghai for Science & Technology, Shanghai, China
chensp@usst.edu.cn

ABSTRACT

Future RFID technologies will go far beyond today's widely-used passive tags. Battery-powered active tags are likely to gain more popularity due to their long operational ranges and richer on-tag resources. With integrated sensors, these tags can provide not only static identification numbers but also dynamic, real-time information such as sensor readings. This paper studies a general problem of how to design efficient polling protocols to collect such real-time information from a subset M of tags in a large RFID system. We show that the standard, straightforward polling design is not energy-efficient because each tag has to continuously monitor the wireless channel and receive $O(|M|)$ tag IDs, which is energy-consuming. Existing work is able to cut the amount of data each tag has to receive by half through a coding design. In this paper, we propose a tag-ordering polling protocol (TOP) that can reduce per-tag energy consumption by more than an order of magnitude. We also reveal an energy-time tradeoff in the protocol design: per-tag energy consumption can be reduced to $O(1)$ at the expense of longer execution time of the protocol. Finally, we apply partitioned Bloom filters to enhance the performance of TOP, such that it can achieve much better energy efficiency without degradation in protocol execution time.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Performance

Keywords

RFID, Energy-efficient, Polling Protocols

1. INTRODUCTION

Traditional barcodes can only be read in close ranges. RFID tags replace barcodes with electronic circuits that can transmit identification numbers wirelessly over a distance. The longer operational range makes them popular in automatic transportation payments, object tracking, and supply chain management [1, 2, 3]. A typical RFID system consists of one or multiple readers and numerous tags. Each tag carries a unique identifier (ID). Tags do not communicate amongst themselves; they communicate directly with the reader.

Passive tags are most widely used today. They are cheap, but do not have internal power sources. They rely on radio waves emitted from the reader for power, and have small operational ranges of a few meters, which seriously limit their applicability. For example, consider a large warehouse in a distribution center of a major retailer, where hundreds of thousands of tagged commercial products are stored. In such an indoor environment, if we use passive tags, hundreds of RFID readers may have to be installed in order to access tags in the whole area, which is not only costly but also causes interference when nearby readers communicate with their tags simultaneously. It is not a good solution to use a mobile reader and walk through the whole area whenever we need information from tags. To automate warehouse management in large scale, a much better choice is to use battery-powered active tags because of their long transmission ranges. The lifetime of these tags is determined by how their battery power is used. Energy conservation must be one of the top priorities in any protocol design that involves active tags.

With richer on-tag resources, active tags are likely to gain more popularity in the future, particularly when their prices drop over time as manufactural technologies are improved and markets are expanded. These tags can be integrated with miniaturized sensors [3, 4, 5]. Not only will they report their IDs, but also they can report dynamic, real-time information about the operation status of the tags or the conditions of the environment.

This paper studies a general problem of how to design efficient polling protocols to collect information from a subset of tags in a large RFID system. For example, consider a large chilled food storage facility, where each food item is attached with a RFID tag that has a thermal sensor. A RFID reader periodically collects temperature readings from tags to check whether any area is too hot (or too cold), which may cause food spoil (or energy waste).[1] Because each area in the facility may be packed with many food items, the temperature readings from these close-by tags are highly redundant. Hence, it is not necessary for the reader to collect information from all tags in the system. The reader may select a subset of tags each time to collect temperature information. In another example, a RFID reader periodically accesses the residual energy levels of on-tag batteries to see if some tags (or their batteries) need to be replaced. If the reader has information about which tags are new and which ones

[1]If a tag reports an abnormal temperature, the reader may instruct the tag to keep transmitting beacons, which guide a mobile signal detector to locate the tag.

MobiHoc'11, May 16-19, 2011, Paris, France.

are old, it may choose to only query the old tags. As we will see later in this paper, it costs less energy to query a smaller number of tags. On the other hand, it is a harder problem to collect information from a *subset* of tags than from *all* tags because the reader has to make sure that tags that are not under query do not transmit — their transmissions will interfere with the transmissions made by tags of interest, causing unnecessary energy waste.

Much existing research focused on designing *ID-collection protocols* that read IDs from all tags in a RFID system [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. In recent years, some interest is shifted to other functions such as estimating the number of tags in a system [1, 12, 16, 17, 18, 19], detecting the missing tags [2, 20], and tag authentication and privacy [21, 22]. The primary performance objective in most papers is to minimize the execution time it takes a protocol to read all tag IDs or perform other functions. Energy efficiency, particularly, how to reduce energy consumption by the tags, is an under-studied subject. There exists prior work on energy-efficient protocols for estimating the number of tags [23], or anti-collision protocols that minimize the energy consumption of a mobile reader [24, 25]. To the best of our knowledge, this paper is the first to investigate energy-efficient polling protocols for collecting information from tags in a large RFID system.

In this paper, we first show that the standard, straightforward polling design is not energy-efficient because each tag has to continuously monitor the wireless channel and receive $O(m)$ tag IDs, which is energy-consuming if the number m of tags that the reader needs to collect information from is large. We then show that a coded polling protocol (CP) is able to cut the amount of data each tag has to receive by half, which means that energy consumption per tag is also reduced by half. This is still far away from our objective of reducing energy consumption to $O(1)$. We propose a novel tag-ordering polling protocol (TOP) that can reduce per-tag energy consumption by more than an order of magnitude when comparing with the coded polling protocol. We also reveal an energy-time tradeoff in the protocol design: per-tag energy consumption can be reduced to $O(1)$ at the expense of longer protocol execution time. Finally, we apply partitioned Bloom filters to enhance the performance of TOP, such that it can achieve much better energy efficiency without degradation in protocol execution time.

The rest of the paper is organized as follows: Section 2 gives the system model and the problem statement. Section 3 describes a baseline protocol. Sections 4 gives a coded polling protocol. Section 5-6 propose a new energy-efficient polling protocol and its enhanced version, respectively. Section 8 presents numerical results. Section 9 discusses the related work. Section 10 draws the conclusion.

2. SYSTEM MODEL AND PROBLEM STATEMENT

We consider a large RFID system using active tags. Each tag carries a unique ID and one or more sensors. It also has the capability of performing certain computations as well as communicating with the RFID reader wirelessly. The reader and the tags transmit with sufficient power such that they can communicate over a long distance. We assume that the RFID reader knows the IDs of all tags in the system by executing an ID-collection protocol, and it has enough power supply.

Let N be the set of tags in the system and $n = |N|$. Let M be a subset of tags, $m = |M|$, and $M \subseteq N$. Our objective is to design efficient polling protocols that collect information from tags in M. A polling protocol may be scheduled to execute periodically. M may change over time so that different subsets of tags are queried. We have two performance objectives. The primary performance objective is to achieve *energy efficiency*. We want to minimize the average amount of energy that a tag spends during one execution of a polling protocol. The energy expenditure by a tag has two components: (1) energy for transmitting its information (e.g., 32 bits) to the reader, and (2) energy for receiving the polling request and other information from the reader. The former is a small, fixed amount of energy that must be spent. The latter is dependent on the protocol design as we will see shortly. It is a variable amount of energy that should be minimized. Simple protocol designs will result in all tags in the system, including those not in M, to be continuously active and unnecessarily receive a large amount of data from the reader for an extended period of time. How to minimize such energy cost is the focus of this paper.

Our secondary performance objective is to reduce *protocol execution time*. RFID systems use low-rate communication channels. For example, in the Philips I-Code system, the rate from a reader to a tag is about 27Kbps and the rate from a tag to a reader is about 53Kbps. Low rates, coupled with a large number of tags, often cause long execution times for RFID protocols. To apply such protocols in a busy warehouse environment, it is desirable to reduce protocol execution time as much as possible.

Communication between the reader and tags is time-slotted. The reader's signal synchronizes the clocks of tags. Let t_{tag} be the length of a time slot during which the reader is able to broadcast a tag ID, and t_{inf} be the length of a time slot during which a tag is able to transmit its information.

3. BASIC POLLING PROTOCOL (BP)

In a standard, straightforward way of designing a polling protocol, we simply let the RFID reader broadcast the tag IDs in M one by one. After it transmits an ID, it waits for a time slot of t_{inf} during which the corresponding tag transmits its information. Each tag continuously listens to the wireless channel. Whenever it receives an ID from the reader, the tag compares the received ID with its own ID. If they match, the tag will transmit its information and then go to sleep until the next scheduled execution of the protocol.

In the above protocol, each tag in M will have to receive $\frac{m}{2}$ IDs on average from the reader before it transmits. Each tag not in M will have to receive all m IDs. The amount of energy spent by a tag in receiving such data grows linearly with respect to m. It takes a constant amount of energy for a tag to receive an ID and another constant amount of energy for it to transmit its information. The energy cost of the whole system is thus $O(nm)$. The protocol execution time is $m(t_{tag} + t_{inf})$.

We use a numerical example to explain the energy cost. Consider a military base that has a large warehouse storing 50,000 weapons, ammunition magazines, and other equipment, which are tagged with RFID sensors. Among them, there are 1,000 sensitive devices, from which a RFID reader needs to access information in order to make sure that they are in good conditions or simply to confirm their presence (against unauthorized removal). Let e_r be the amount of energy a tag spends in receiving an ID and e_s be the amount of energy a tag spends in transmitting its information. The total energy consumed by all tags for transmitting is $1,000e_s$, and the total energy consumed by all tags for receiving is about $50,000,000e_r$. Even though e_r may be smaller than e_s, the total amount of energy spent by tags in receiving can be much greater than the amount spent in transmitting.

4. CODED POLLING PROTOCOL

We show that a coded polling protocol (CP) [26] is able to reduce

the amount of data each tag has to receive by half. The protocol assumes that each tag ID carries an *identification number* and a CRC (*cyclic redundancy code*) for error detection. This requirement is satisfied by the EPCglobal Gen-2 standard, where each 96-bit tag ID contains a CRC checksum. The CRC is computed based on the identification number and a generator. When a tag receives an ID from a wireless channel, it computes a CRC based on the received identification number and then compares the result with the received CRC. If they are the same, we say the ID contains a *valid* CRC.

CRC has the following property: If x and y are two tag IDs with valid CRCs, then $x \oplus y$ also has a valid CRC. The same property does not hold for $x \oplus \hat{y}$, where $\hat{y}$ contains the same bits in y but in the reverse order. For example, if $y = 10110$, then $\hat{y} = 01101$. We call $\hat{y}$ the *reversal* of y.

In the coded polling protocol, the RFID reader first arranges the IDs in M in pairs. Each pair consists of two IDs that are arbitrarily selected from M. Consider an arbitrary pair, x and y, which are called each other's *paring ID*. We define the *polling code* of the pair as $c = x \oplus \hat{y}$.

Instead of sending out the IDs in M one after another, the reader broadcasts the polling code of each pair one after another. After each broadcast of a polling code $c = x \oplus \hat{y}$, the reader waits for two time slots, during which tag x and tag y will transmit. More specifically, when an arbitrary tag z receives the polling code c, it first computes $z \oplus c$, and checks whether the CRC in the reversal of $z \oplus c$ is valid. If it is, the tag will transmit its information. Otherwise, the tag computes $\hat{z} \oplus c$, and checks whether the CRC in $\hat{z} \oplus c$ is valid. Again, if it is valid, the tag will transmit. Otherwise, the tag will not transmit. We show that only tag x and tag y will transmit.

First, consider the case of $z = x$. The tag first computes $z \oplus c = x \oplus x \oplus \hat{y} = \hat{y}$. The reversal of $\hat{y}$ is y. The CRC in any tag ID (including y) is valid. Hence, tag x will transmit. Moreover, it now knows its pairing ID, y. If x is greater than y, the tag will transmit in the first slot after receiving the polling code; otherwise, it will transmit in the second slot.

Second, we consider the case of $z = y$. The tag first computes $y \oplus c = y \oplus x \oplus \hat{y}$. Its reversal is likely to have an invalid CRC; the chance for an arbitrary number to contain a valid CRC is very small. Then, the tag computes $\hat{z} \oplus c = \hat{y} \oplus x \oplus \hat{y} = x$, which contains a valid CRC. Consequently, y will transmit. Since it now knows its pairing ID, x, it also knows in which slot it should transmit.

Finally, consider the case of $z \neq x$ and $z \neq y$. The tag computes the reversal of $z \oplus c = z \oplus x \oplus \hat{y}$ and then computes $\hat{z} \oplus c = \hat{z} \oplus x \oplus \hat{y}$. Both of them are likely to have invalid CRCs.

A minor problem is that $y \oplus c$ in the second case and $z \oplus c$ or $\hat{z} \oplus c$ in the third case still have a small probability to contain a valid CRC. However, the reader can easily prevent this from happening. It knows all tag IDs. It can precompute all polling codes and check whether a valid CRC happens in the above cases by chance when it is not supposed to. If this is true for a pair of tags, x and y, the reader must break up the pair, and use them to form new pairs with other IDs in M. Such an approach is effective because the probability for this to happen is exceedingly small when CRC is sufficiently long.

Because each polling code represents two tag IDs, the number of polling codes in CP is $\frac{m}{2}$. Hence, when comparing with the basic polling protocol, CP reduces the number of broadcasts made by the reader by half, and it also reduces the amount of data that each tag has to receive by half. This not only saves energy for tags, but also reduces the protocol execution time to $\frac{m}{2} t_{tag} + m t_{inf}$.

5. TAG-ORDERING POLLING PROTOCOL (TOP)

Although CP is more efficient, the expected amount of energy that each tag spends in receiving remains $O(m)$. In this section, we propose a new tag-ordering polling protocol that reduces such energy cost to $O(1)$.

5.1 Motivation

In the basic polling protocol, a RFID reader broadcasts m IDs in time slots of length t_{tag}. All tags must continuously monitor the wireless channel in order to know whether their own IDs are in the broadcast. In CP, the reader broadcasts $\frac{m}{2}$ polling codes also in time slots of length t_{tag}. Again, all tags must continuously monitor the wireless channel. They have to keep receiving and processing the polling codes. Each tag in the basic protocol has to receive up to m IDs. Even though CP is more efficient, a tag still has to receive up to $\frac{m}{2}$ codes.

We want to remove the necessity for any tag to keep monitoring the wireless channel. Ideally, a tag should stay in an energy-conserving standby mode for most of time, and only wake up at the right time slot to receive information about itself, such as whether it is polled and, if so, when it should transmit. To further reduce the amount of data that tags have to receive, we let the reader broadcast a so-called *reporting-order* vector V, instead of IDs in M. Each ID in M is mapped to a bit in V through a hash function; the bit is set as one to encode the ID in the vector. A tag only needs to check a specific bit in V at a location determined by the hash of its ID. This bit is called the *representative bit* of the tag. If its value is one, the tag is polled by the reader for reporting, i.e., the tag belongs to M; if its value is zero, the tag is not polled. The vector V also carries information about the order in which the polled tags will report their data. Each bit whose value is one in V represents a polled tag. If a tag finds that there are i ones in V preceding its representative bit, it knows that it should be the $(i+1)$th tag in M to report its information. With such an ordering, it becomes possible for tags in M to report at different times and avoid collision.

However, this basic idea has two problems. First, there should be at least m bits in V to encode m IDs in M. The energy cost of receiving V remains $O(m)$. How can a tag find out the number of ones in V preceding its representative bit without having to receive the whole vector? Second, hash collision causes two issues. If a tag not in M is hashed to the same bit in V as a tag in M does, it will find its representative bit to be one, causing false positive. If two tags in M are mapped to the same bit in V, they will transmit at the same time, causing report collision. In the rest of this section, we design a new tag-ordering polling protocol (TOP) to solve these problems. It consists of three phases: *ordering phase*, *polling phase*, and *reporting phase*. In the ordering phase, the reader broadcasts the vector V so that each tag knows whether it is polled and where it is located in the reporting order. The polling phase resolves the issues of false positive and report collision. Finally, in the reporting phase, tags in M report their information in the order defined by V without collision.

5.2 Protocol Description

5.2.1 Ordering phase

The RFID reader does not broadcast any IDs or indices. It only broadcasts the reporting-order vector, V. If V cannot fit in one time slot of length t_{tag}, the reader breaks the vector into segments and broadcasts each segment in a time slot of t_{tag}. In addition, the reader also broadcasts the vector size v.

Knowing the vector size, a tag t is able to hash its ID and find out the location of its representative bit in V. Because the segment size is fixed, t also knows which segment its representative bit belongs

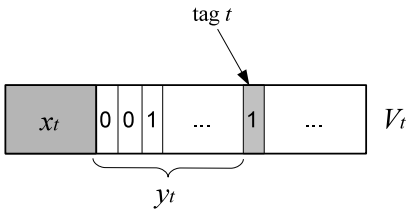

Figure 1: V_t is the representative segment of tag t, x_t is the total number of ones in all previous segments, and y_t is the number of ones in V_t that precede tag t's representative bit. I_t is the position of t in the reporting order. $I_t = x_t + y_t$.

to. This segment, denoted as V_t, is called the *representative segment* of tag t. A tag will stay in the standby mode and be active only when receiving its representative segment.

If a tag finds that its representative bit is zero, it knows for sure that it is not a member in M. If a tag finds that its representative bit is one, it may be a member in M or a non-member that is mapped to a bit which a member in M is also mapped to. The latter case causes *false positive*. **Because the reader knows all IDs in the system, it can pre-compute the set F of non-member tags that cause false positive.**

When the reader broadcasts any segment of V, it includes in the same time slot *the total number of ones in the previous segments*. For an arbitrary tag t, let I_t be the number of ones in V preceding the representative bit of t. When tag t receives V_t, it can computes I_t as the sum of (a) the number of ones in the previous segments and (b) the number of ones in V_t before its representative bit. See Figure 1 for illustration. As we will see later, the value of I_t specifies when tag t will transmit during the reporting phase.

If two tags in M are mapped to the same bit in V, they will have the same I_t value and thus transmit at the same time during the reporting phase, causing collision. **Because the reader has all IDs in M, it knows exactly which tags will be mapped to the same bit.** This makes it easy to resolve collision. The reader simply removes all but one tag that are mapped to a bit, and puts them in a set C. These tags, together with tags in F, will not participate in the reporting phase. They are handled separately in the polling phase.

5.2.2 Polling Phase

In this phase, the reader issues two types of polling requests. For each tag in C, it sends a *positive polling request*. For each tag in F, it sends a *negative polling request*. To distinguish these two types, the reader must transmit a one-bit flag together with a tag ID in each request, specifying whether the polling is positive or negative and which tag is polled.

Tags that find their representative bits to be ones in the previous phase must continuously listen to the channel during the polling phase. After sending a positive request, the reader waits for a time slot to receive information. The tag that finds its ID in the request will transmit its information in this slot. This tag, which belongs to C, will not participate in the reporting phase. After sending a negative request, the reader does not wait before sending out the next request. The tag that finds its ID in a negative request knows that it must belong to F and hence should not further participate in the protocol execution.

The total number of polling requests is $|F|+|C|$. By choosing an appropriate size for the reporting-order vector, we can make sure that $|F|+|C| = O(1)$ (see Section 5.3). Note that only tags in M and F have to listen to the channel in this phase. Tags in $N-M-F$, which may contain the majority of tags in the system, have already known that they do not belong to M and thus do not need to participate in the protocol execution.

5.2.3 Reporting phase

A tag participates in the reporting phase only if it satisfies the following two conditions: (1) it finds that its representative bit is one in the ordering phase, and (2) it does not find its ID in the requests of the polling phase.

The reporting phase consists of $m - |C|$ time slots. In each time slot, one tag in $M - C$ transmits its information. Recall that each tag in M learns its index in the reporting order during the ordering phase. The tag will transmit in the reporting phase at the time slot of the same index.

5.2.4 Timing

Before executing the protocol, the RFID reader uses its broadcasting signal to synchronize the clocks of the tags. The reader computes the vector V and breaks it into segments. Suppose each time slot of length t_{tag} can carry 96 bits. We may set the segment size to be 80 bits and use the remaining 16 bits to carry the total number of ones in the previous segments.[2] The reader is able to compute the execution time T_1 of the ordering phase, which is the number of segments multiplied by t_{tag}.

Since the reader knows all IDs in the system, it can precompute the set F of tags that cause false positive and the set C of tags that should not participate in the reporting phase in order to avoid collision. Based on F and C, the reader can compute the execution time T_2 of the polling phase, which is $|F| \times t_{tag} + |C| \times (t_{tag} + t_{inf})$.

Suppose all tags wake up at each scheduled execution of the protocol. The reader computes and broadcasts the values of T_1 and T_2 right before the ordering phase, so that the tags know when each phase of the protocol will begin. They will remain in the standby mode unless they have to receive their representative segments, participate in the polling phase, or transmit their information in the reporting phase.

If the system requires on-demand polling of tag information instead of periodic execution, there are two possible solutions to wake the tags up in the first place. The first one is "pseudo-on-demand" polling, where tags still wake up periodically, but the reader only issues the polling request when needed. The second approach is to attach a wake-up circuit to each tag, and use the two-stage wake-up scheme proposed in [27] to activate the tags. In this approach, tags responde almost immediately to the polling event. However, the wake-up circuit requires the reader to be close enough so that the radio power is strong enough to trigger the wake-up event. As a result, we may have to deploy extra readers to cover all the tags.

5.3 Performance Analysis

5.3.1 Energy Cost

We show how to configure TOP such that the energy cost per tag is $O(1)$. The energy cost of a tag has four components: (1) receiving v, T_1 and T_2, (2) receiving a segment of V in the ordering phase, (3) listening to the channel during the polling phase, and (4) transmitting information in a slot at the reporting phase (or at the polling phase if the tag is in C). The first two components incur small, constant energy expenditure to every tag in the system. The fourth

[2]Using 16 bit to carry the number of ones in previous segments will limit the value of m to (0, 65,535]. To get rid of this limitation, we can use $\lceil \log_2 m \rceil$ bits instead and broadcast the value of $\lceil \log_2 m \rceil$ to tags at the beginning of protocol. However, for the sake of simplicity, we use 16 bits in this paper to help demonstrate the main idea.

component also incurs small, constant energy cost, but only to the tags in M. The third component incurs energy cost only to tags in F and M. In the worse case, a tag has to listen to all $|C| + |F|$ polling requests from the reader. Suppose it takes one unit of energy to receive a polling request. The total energy cost of a tag, denoted as Ω, is

$$\Omega \leq |C| + |F| + O(1). \quad (1)$$

We treat $|C|$ and $|F|$ as random variables and derive their expected values. Let v be the number of bits in the reporting-order vector V. Let b_i be the value of the ith bit in V, $0 \leq i < v$. For each tag in M, the reader maps it to a random bit in V and sets the bit to one. After encoding all m tags in V, the probability for b_i to be one is

$$Prob\{b_i = 1\} = 1 - (1 - \frac{1}{v})^m \approx 1 - e^{-m/v}.$$

The bits, b_0, b_2, ..., b_{v-1}, are independent of each other. Thus, the expected number of ones in V is $\sum_{i=1}^{v} Prob\{b_i = 1\}$. The value of $|C|$ is equal to m subtracted by the number of ones in V. Hence, we have

$$E(|C|) = m - \sum_{i=1}^{v} Prob\{b_i = 1\} \approx m - v(1 - e^{-m/v}). \quad (2)$$

A tag not in M will cause false positive when its representative bit is one. The probability for this to happen is $Prob\{b_i = 1\}$. Hence,

$$E(|F|) = (n - m)Prob\{b_i = 1\} \approx (n - m)(1 - e^{-m/v}) \quad (3)$$

Both $E(|C|)$ and $E(|F|)$ are monotonically decreasing functions of v. We show that $E(|C|) = O(1)$ if v is sufficiently large. Let $v = \frac{m^2}{2}$. From Taylor expansion, we know that

$$\begin{aligned} 1 - e^{-m/v} &= \frac{m}{v} - \frac{1}{2!}(\frac{m}{v})^2 + \frac{1}{3!}(\frac{m}{v})^3 - \frac{1}{4!}(\frac{m}{v})^4 ... \\ &\geq \frac{m}{v} - \frac{1}{2!}(\frac{m}{v})^2 \end{aligned}$$

Applying it to (2), we have

$$E(|C|) = m - v(1 - e^{-m/v}) \leq \frac{1}{2!}\frac{m^2}{v} = 1. \quad (4)$$

Next we show that $E(|F|) = O(1)$ if v is sufficiently large. If $n = m$, $E(|F|) = 0$. Now assume $n > m$. Let $v = -\frac{m}{\ln(1-\frac{1}{n-m})}$. It can be transformed into

$$(n - m)(1 - e^{-m/v}) = 1.$$

The left side is $E(|F|)$. Therefore, if we choose $v = \max\{\frac{m^2}{2}, -\frac{m}{\ln(1-\frac{1}{n-m})}\}$, we have

$$E(\Omega) \leq E(|C|) + E(|F|) + O(1) \leq 1 + 1 + O(1) = O(1).$$

We conclude that TOP can be configured such that the expected energy cost per tag is $O(1)$. As we will see shortly, the protocol execution time increases when v becomes too large. To strike a balance between energy cost and protocol execution time, we may choose a value of v much smaller than $\max\{\frac{m^2}{2}, -\frac{m}{\ln(1-\frac{1}{n-m})}\}$. In Section 8, we use simulations to study the performance of TOP under practical values of v. For example, when $v = 24m$, the amount of data that a tag receives in TOP is more than an order of magnitude smaller than what a tag has to receive in CP.

We characterize the energy cost in the polling phase by counting the amount of data (in Kilobits) that a tag has to receive. Numerical results are shown in the first plot of Figure 2, where $n = 50,000$ and $m = 5,000, 10,000$, or $25,000$, corresponding to three curves in the plot. Clearly, as v increases, the energy cost decreases.

5.3.2 Execution Time

The protocol execution time also consists of four components. To begin with, it takes the reader a small, constant time to broadcast v, T_1 and T_2. The time for the ordering phase is $\frac{v}{l}t_{tag}$, where l is the segment size. The time for the polling phase is $|F| \times t_{tag} + |C| \times (t_{tag} + t_{inf})$. The time for the reporting phase is $|M - C| \times t_{inf}$. Hence, the total execution time is $T = (\frac{v}{l} + |F| + |C|)t_{tag} + m \times t_{inf} + O(1)$.

From (2) and (3), the expected protocol execution time is

$$\begin{aligned} E(T) &= \Big[\frac{v}{l} + (n - m)(1 - e^{-m/v}) + m \\ &\quad - v(1 - e^{-m/v})\Big]t_{tag} + m \cdot t_{inf} + O(1) \\ &\approx \Big[\frac{v}{l} + \frac{(n - m)m}{v}\Big]t_{tag} + m \cdot t_{inf} + O(1) \quad (5) \end{aligned}$$

The second plot of Figure 2 presents the protocol execution time (excluding the constant $O(1)$) when $n = 50,000$, $m = 5,000, 10,000$, or $25,000$, $t_{tag} = 3297\mu s$, and $t_{inf} = 906\mu s$; see Section 8 for how they are determined. Interestingly, as v increases, the execution time first decreases and then increases. We can find the optimal value of v that minimizes the execution time from $\frac{\delta E(T)}{\delta v} = 0$.

Combining the results in the first and second plots, we can figure out the *tradeoff relation* between energy cost and protocol execution time, which is presented in the third plot. As v becomes large, the energy cost decreases at the expense of increased execution time.

6. ENHANCED TAG-ORDERING POLLING PROTOCOL (ETOP)

6.1 Motivation

If we do not want to significantly increase execution time, we cannot choose a large value for v. In this case, we must find other means to lower energy cost. The key is to reduce the number of IDs that have to be transmitted in the polling phase. Namely, we should reduce the number of tags in F and C. Let's first focus our discussion on false positive. Consider an arbitrary tag $t \notin M$. Its representative segment is V_t. Let q be the number of tags in M that are also mapped to V_t. False positive occurs if t and one of those q tags have the same representative bit. The probability for this to happen is $1 - (1 - \frac{1}{l})^q$, where l is the number of bits in V_t.

To further reduce the false-positive probability, we can implement each segment of V as a Bloom filter [28, 29]. The reader uses multiple hash functions to map each tag to $k(> 1)$ *representative bits* in V, instead of just one in TOP. More specifically, for each member $t' \in M$, the reader first maps it to a representative segment $V_{t'}$ through a hash function whose range is $[0, \frac{v}{l})$. Then the reader further maps t' to k representative bits in $V_{t'}$ and set them to ones.

After all members in M are encoded in the segments of V, the reader broadcasts the segments in the ordering phase. A tag t only listens for its representative segment V_t and then checks its representative bits. If any representative bit is zero, the tag can not be in M. If all representative bits are ones, the tag may be a member in M or a false positive. In the case of false positive, even though the tag does not belong to M, every one of its representative bits is set because it is also a representative bit of a member tag in M. The probability for this to happen is $(1 - (1 - \frac{1}{l})^{kq})^k$, where q is the number of tags in M whose representative segments are also V_t. For example, if $l = 80$, $k = 3$, and $q = 2$, the false-positive probability is just 3.8×10^{-4}, much lower than $1 - (1 - \frac{1}{l})^q = 2.5 \times 10^{-2}$ in TOP under the same parameters.

Bloom filters can reduce the false-positive probability. But it is

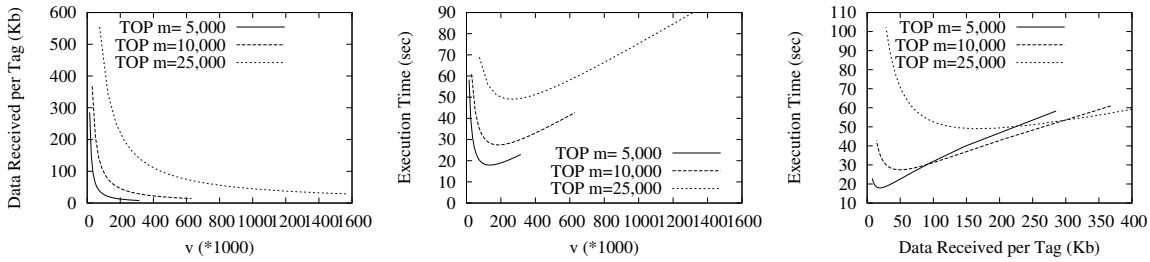

Figure 2: First plot: Energy cost per tag with respect to v. Second plot: Protocol execution time with respect to v. Third plot: Energy-time tradeoff controlled by v.

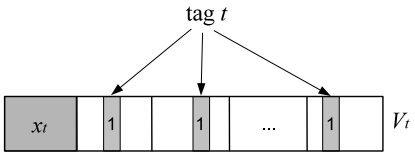

Figure 3: V_t is the representative segment of tag t. V_t is evenly divided into k partitions, each having $\lfloor \frac{l}{k} \rfloor$ bits. Tag t has one representative bit in every partition.

more difficult to use them to carry the reporting order, based on which the tags will take turn to transmit during the reporting phase. In TOP, we use the number of ones that precede the representative bit of a tag to determine the tag's position in the reporting order. Bloom filters use multiple representative bits to encode each member. The representative bits of different members may overlap in an arbitrary way. Hence, we cannot simply use all bits whose values are ones to represent tags in M because there is no one-to-one mapping between them.

In the following, we design an *enhanced tag-ordering polling protocol* (ETOP) to solve the above problem. ETOP uses *partitioned Bloom filters*, which not only reduce false positive and encode the reporting order, but also reduce $|C|$ as well as overall execution time of the protocol.

6.2 Protocol Description

The main difference between ETOP and TOP is that ETOP implements each segment of V as a partitioned Bloom filter instead of a simple bit array. When we describe the protocol of ETOP, we focuses on the difference while omitting the details that it shares in common with TOP.

In a partitioned Bloom filter, the l bits of a segment are evenly divided into k partitions. Each partition has $\lfloor \frac{l}{k} \rfloor$ bits. See Figure 3 for illustration. For every member tag t in M, the reader applies a hash function on its ID to obtain a number of hash bits. The reader uses $\lceil \log_2 v \rceil$ hash bits to map t to a representative segment V_t, and then uses $k\lceil \log_2 \frac{l}{k} \rceil$ hash bits to further map t to one representative bit in every partition of the segment. Like a classical Bloom filter, the partitioned Bloom filter sets k representative bits for each encoded member; unlike a classical Bloom filter, a partitioned Bloom filter spreads the k representative bits in k different partitions.

After receiving its representative segment, a tag checks the k representative bits to determine if it is a member in M. False positive cases are handled by the reader in the polling phase as usual.

How does a tag t know its position in the reporting order? First we consider the reporting order among tags that are encoded in the same segment V_t. Since every tag has exactly one representative bit in each partition of V_t, we may be able to use one of the partitions to carry the order information. In other words, if there is a partition P^* whose number of ones is equal to the number of tags encoded in V_t, we know that there must be a one-to-one mapping between these tags and the '1' bits in P^*. We can use the order of '1' bits in P^* as the reporting order of the corresponding tags. We will explain shortly how the reader makes sure that such a partition exists. When the reader sends out V_t, in the same time slot it also sends the total number x_t of tags that are encoded in all previous segments of V. The position of tag t in the reporting order can be computed from x_t and the information in P^*, which we will further explain later.

We want to make sure that any segment of V always has a partition whose number of ones is equal to the number of tags encoded in the segment. The reader has to do some extra work. After encoding all tags in M, the reader examines the partitions one by one for each segment. If there is not such a partition, the reader removes an encoded tag and places it in the set C, which will be explicitly polled in the polling phase. The reader keeps removing tags until it finds a partition that satisfies the above requirement. Note that the requirement is always satisfied when the number of tags encoded in a segment is one.

After receiving its representative segment V_t, a tag $t \in M$ computes its position in the reporting order as follows: It finds out a partition P^* in V_t that has the largest number of ones. This partition must have a one-to-one mapping between '1' bits and encoded tags. Let y_t be the number of ones in P^* that precedes the representative bit of t. The tag computes its position in the reporting order as $y_t + x_t$. Recall that x_t is the number of tags that are encoded in the previous segments. It is received together with V_t in the same time slot.

The polling phase and the reporting phase of ETOP are identical to their counterparts in TOP.

6.3 Performance Analysis

6.3.1 Energy Cost

We show that ETOP can be configured such that the energy cost per tag is $O(1)$. The actual energy cost will be studied by simulations in Section 8.

ETOP has the same upper bound formula on per-tag energy cost as TOP, shown in (1), but with different values of $|C|$ and $|F|$. In the following, we will derive $|C|$ and $|F|$ for ETOP. Let m_i be the number of tags in M that are encoded in the ith segment, $0 \leq i < \frac{v}{l}$. Each tag in M has a probability of $\frac{l}{v}$ to be mapped to the ith segment. Hence, m_i follows a binomial distribution $Bino(m, \frac{l}{v})$.

$$Prob\{m_i = x\} = \binom{m}{x} (\frac{l}{v})^x (1 - \frac{l}{v})^{m-x} \quad (6)$$

Let C_i be a subset of C, containing the tags that are removed from the ith segment. We know the following facts: (1) When $m_i = 0$, $|C_i| = 0$. (2) When $m_i = 1$, $|C_i| = 0$. (3) When $m_i \geq 1$, $|C_i| \leq m_i - 1$. Hence, we must have

$$E(|C_i|) < (m_i - 1) \cdot \Big(1 - Prob\{m_i = 0\} - Prob\{m_i = 1\}\Big)$$
$$= (m_i - 1) \cdot \big(1 - (1 - \frac{l}{v})^m - \frac{ml}{v}(1 - \frac{l}{v})^{m-1}\big).$$

Since $(1 - \frac{l}{v})^m > 1 - \frac{ml}{v}$, we have

$$E(|C_i|) < \frac{m_i(m-1)^2 l^2}{v^2} < \frac{m_i m^2 l^2}{v^2}.$$

$|C|$ is the sum of all $|C_i|$s, $0 \leq i < \frac{v}{l}$. We know $\sum_{i=1}^{v/l} m_i = m$. So,

$$E(|C|) = \sum_{i=1}^{v/l} E(|C_i|) < m\frac{m^2 l^2}{v^2} = \frac{m^3 l^2}{v^2}.$$

If we let $v = v_1 = \sqrt{m^3 l^2}$, $E(|C|) < 1$.

Consider an arbitrary tag not in M. Without loss of generality, suppose it is mapped to the ith segment. In any partition of the segment, the probability for it to share a representative bit with a tag in M is $1 - (1 - \frac{k}{l})^{m_i}$. The probability for that to happen in all partitions is $[1 - (1 - \frac{k}{l})^{m_i}]^k$. Hence, the probability for the tag to cause false positive, denoted as p_f is

$$p_f = \sum_{q=1}^{m} Prob\{m_i = q\}\big[1 - (1 - \frac{k}{l})^q\big]^k$$
$$< (1 - Prob\{m_i = 0\})\big[1 - (1 - \frac{k}{l})^m\big]^k.$$
$$\approx (1 - e^{-lm/v})(1 - e^{-km/l})$$

The expected valus of $|F|$ is

$$E(|F|) = (n - m) \cdot p_f$$
$$< (n - m)(1 - e^{-lm/v})(1 - e^{-km/l}) \quad (7)$$

If we let $v = v_2 = -\frac{ml}{\ln(1 - \frac{1}{(n-m)(1-e^{-km/l})})}$ and apply it to (7), we have $E(|F|) < 1$. Now, if we choose $v = \max\{v_1, v_2\}$, the expected energy cost $E(\Omega) \leq E(|C|) + E(|F|) + O(1) < 1 + 1 + O(1) = O(1)$. Therefore, ETOP can also be configured such that the energy cost per tag is $O(1)$.

6.3.2 Execution Time

Following the same analysis as in Section 5.3.2, it is easy to see that ETOP has the same formula for protocol execution time as TOP: $T = (\frac{v}{l} + |F| + |C|)t_{tag} + m \times t_{inf} + O(1)$, but the values of $|C|$ and $|F|$ are different. Our simulation results in Section 8 show that ETOP has smaller execution time than TOP.

7. CHANNEL ERROR

Channel error may corrupt the data exchanged between the reader and tags. For example, if a negative polling request is corrupted, the tag that is not supposed to participate in the reporting phase will transmit and cause collision in the reporting phase. A segment of V sent from the reader may be corrupted so that tags encoded in this segment will not report their information. There exists other scenarios of corruption in the execution of TOP or ETOP. They cause two effects: 1) A tag in M does not transmit its information in the slot when it is supposed to transmit, and 2) it transmits but collides with another tag that is not supposed to transmit in the slot. To detect these cases, when a tag transmits, we require it to include a CRC checksum that is computed from the concatenation of the information bits and the tag's ID. When the reader expects information from a tag in a time slot, if the slot turns out to be empty or the data received in the slot do not carry a correct CRC, the reader knows that information from the tag is not correctly received. At the end of the protocol, all missed information can be retrieved by polling the tags directly.

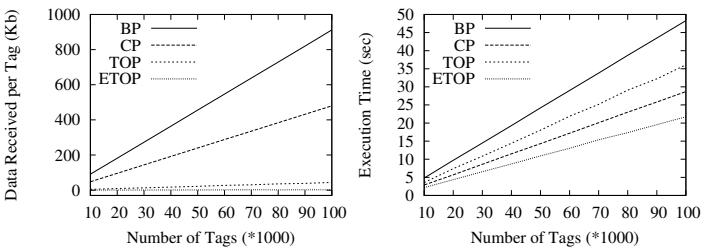

Figure 4: Energy and time comparison. Parameters: $m = 0.1n$, $v = 24m$ for TOP and ETOP. Note that the horizontal '0' line is not at the bottom in order to make the ETOP curve visible.

8. SIMULATION RESULTS

In this section, we evaluate the performance of our new protocols, the tag ordering polling protocol (TOP) and the enhanced tag ordering polling protocol (ETOP). We compare them with the basic polling protocol (BP) and the coded polling protocol (CP). Our evaluation uses two performance metrics: (1) the average number of bits that each tag has to receive during the protocol execution, and (2) the overall execution time.

We only consider energy consumption of tags in receiving information for two reasons. First, this is the major, variable portion of the energy cost per tag. As we will see shortly, each tag may have to receive hundreds of thousands of bits during protocol execution, whereas it only sends a small, fixed amount, e.g., 32 bits. Second, the energy cost for tags in M to transmit their information is the same for all protocols. Omitting them does not affect the comparison.

We use the following parameters to configure the simulation: each tag ID is 96 bits long, information reported from a tag to the reader is 32 bits long, and each segment in ETOP is 80 bits long and divided into 4 partitions, i.e. $k = 4$. The transmission time is based on the parameters of the Philips I-Code specification [30]. The rate from a tag to the reader is 53Kb/sec; it takes $18.88\mu s$ for a tag to transmit one bit. Any two consecutive transmissions (from the reader to tags or vice versa) are separated by a waiting time of $302\mu s$. The value of t_{inf} is calculated as the sum of a waiting time and the time for transmitting the information, which is $18.88\mu s$ multiplied by the length of the information. For 32-bit information, $t_{inf} = 906\mu s$. The transmission rate from the reader to tags is 26.5Kb/sec; it takes $37.76\mu s$ for the reader to transmit one bit. The value of t_{tag} is calculated as the sum of a waiting time and the time for transmitting a 96-bit ID. The result is $3927\mu s$.

8.1 Varying number n of tags

We first vary the number n of tags in the system from 10,000 to 100,000. We set $v = 24m$ and $m = 0.1n$, i.e., 10% of all tags are selected by the reader to report information. Figure 4 compares four protocols in terms of energy cost and protocol execution time. The left plot shows energy costs. TOP and ETOP reduce energy consumption by one or multiple orders of magnitude. For example,

when $n = 100,000$, per-tag energy cost in TOP is 9.4% of the cost in CP, and 5.0% of the cost in BP. Per-tag energy cost in ETOP is just 0.52% of the cost in CP, and 0.28% of the cost in BP. The right plot shows the execution time comparison. TOP requires 25% less time than BP, but 27% more time than CP. ETOP requires 55% less time than BP and 24% less time than CP.

In summary, CP reduces both energy cost and execution time nearly by half when comparing with BP. TOP makes great improvement over CP in terms of energy cost, but has modestly higher execution time. ETOP considerably outperforms CP in terms of both energy cost and execution time.

8.2 Varying size v of reporting-order vector

Next, we show how the value of v influences the performance of TOP and ETOP. We set $n = 50,000$ and $m = 5,000$, $10,000$, or $25,000$. We vary v from $4m$ to $64m$ and use simulation to find energy cost per tag and protocol execution time. Figure 5 shows the simulation results. The first two plots present the average amount of data each tag receives in TOP and ETOP, respectively. The curves match the theoretical results we have given in Section 5.3. When v is reasonably large, e.g., $v \geq 7m$, ETOP consumes less energy than TOP. The third and forth plots present the protocol execution time of TOP and ETOP, respectively. ETOP also requires less time than TOP when $v \geq 7m$.

9. RELATED WORK

Much existing work on RFID systems is to design anti-collision ID-collection protocols, which read IDs from all the tags in the system. They mainly fall into two categories. One is *ALOHA-based* [10, 11, 12, 13, 14, 15], and the other is *Tree-cased* [6, 7, 8, 9]. The *ALOHA-based protocols* work as follows: The reader broadcasts a query request. With a certain probability, each tag chooses a time slot in the current frame to transmit its ID. If there is a collision, the tag will continue participating in the next frame. This process repeats until all tags are identified successfully.

The *tree-based protocols* organize all IDs in a tree of ID prefixes [6, 7, 8]. Each in-tree prefix has two child nodes that have one additional bit, '0' or '1'. The tag IDs are leaves of the tree. The RFID reader walks through the tree. As it reaches an in-tree node, it queries for tags with the prefix represented by the node. When multiple tags match the prefix, they will all respond and cause collision. Then the reader moves to a child node by extending the prefix with one more bit. If zero or one tag responds (in the one-tag case, the reader receives an ID), it moves up in the tree and follows the next branch. Another type of tree-based protocols tries to balance the tree by letting the tags randomly pick which branches they belong to [6, 9, 31].

Other work designs time-efficient protocols to estimate the number of tags in a large RFID system [1, 16, 17, 18]. Kodialam and Nandagopal [16] propose a probabilistic model to estimate the number of tags. A follow-up work can be found in [1]. Qian et al. [18] present the Lottery-Frame scheme (LoF) for a multiple-reader scenario. The work by Li et al. focuses on energy efficiency, instead of time efficiency, on their solutions for estimating the number of tags in a system [23].

Tan, Sheng and Li [2] design a Trust Reader Protocol (TRP) to detect the missing-tag event with probability α when the number of missing tags exceeds m, where α and m are system parameters. Li, Chen and Ling [20] propose a series of protocols to exactly identify the IDs of the missing tags. Yang et al. [21] design a Single Echo based Batch Authentication (SEBA) to authenticate a batch of tags.

10. CONCLUSION

In this paper, we propose two energy-efficient polling protocols, TOP and ETOP, for large-scale RFID systems. These protocols are designed to collect real-time information from a subset of tags in the system. Our primary objective is to lower energy consumption by tags in order to extend their lifetime. The new protocols can be configured to achieve $O(1)$ energy cost per tag. Performance tradeoff between energy cost and execution time can be made by controlling the size of the reporting-order vector. Simulation results show that the new protocols are able to cut energy cost by more than an order of magnitude, when comparing with other protocols.

11. ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grant CPS-0931969.

12. REFERENCES

[1] M. Kodialam, T. Nandagopal, and W. Lau, "Anonymous Tracking using RFID tags," *Proc. of IEEE INFOCOM*, 2007.

[2] C. Tan, B. Sheng, and Q. Li, "How to Monitor for Missing RFID Tags," *Proc. of IEEE ICDCS*, 2008.

[3] L. M. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor Location Sensing using Active RFID," *ACM Wireless Networks (WINET)*, vol. 10, no. 6, November 2004.

[4] A. Ruhanen, M. Hanhikorpi, F. Bertuccelli, A. Colonna, W. Malik, D. Ranasinghe, T. S. Lopez, N. Yan, and M. Tavilampi, "Sensor-enabled RFID Tag Handbook," *BRIDGE, IST-2005-033546*, January 2008.

[5] M. Miura, S. Ito, R. Takatsuka, T. Sugihara, and S. Kunifuji, "An Empirical Study of an RFID Mat Sensor System in a Group Home," *Journal of Networks*, vol. 4, no. 2, April 2009.

[6] J. Myung and W. Lee, "Adaptive Splitting Protocols for RFID Tag Collision Arbitration," *Proc. of ACM MOBIHOC*, 2006.

[7] N. Bhandari, A. Sahoo, and S. Iyer, "Intelligent Query Tree (IQT) Protocol to Improve RFID Tag Read Efficiency," *Proc. of IEEE ICIT*, 2006.

[8] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min, "Evaluating and Optimizing Power Consumption of Anti-collision Protocols for Applications in RFID Systems," *Proc. of ISLPED*, 2004.

[9] "Information Technology – Radio Frequency Identification for Item Management Air Interface – Part 6: Parameters for Air Interface Communications at 860-960 MHz," *Final Draft International Standard ISO 18000-6*, November 2003.

[10] S. Lee, S. Joo, and C. Lee, "An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification," *Proc. of IEEE MOBIQUITOUS*, 2005.

[11] J. R. Cha and J. H. Kim, "Dynamic Framed Slotted ALOHA Algorithms using Fast Tag Estimation Method for RFID Systems," *Proc. of IEEE CCNC*, 2006.

[12] H. Vogt, "Efficient Object Identification with Passive RFID Tags," *Proc. of IEEE PerCom*, 2002.

[13] V. Sarangan, M. R. Devarapalli, and S. Radhakrishnan, "A Framework for Fast RFID Tag Reading in Static and Mobile Environments," *The International Journal of Computer and Telecommunications Networking*, vol. 52, no. 5, 2008.

[14] B. Zhen, M. Kobayashi, and M. Shimizu, "Framed ALOHA for Multiple RFID Objects Identification," *IEICE Transactions on Communications*, 2005.

[15] B. Sheng, Q. Li, and W. Mao, "Efficient Continuous Scanning in RFID Systems," *Proc. of IEEE INFOCOM*, 2010.

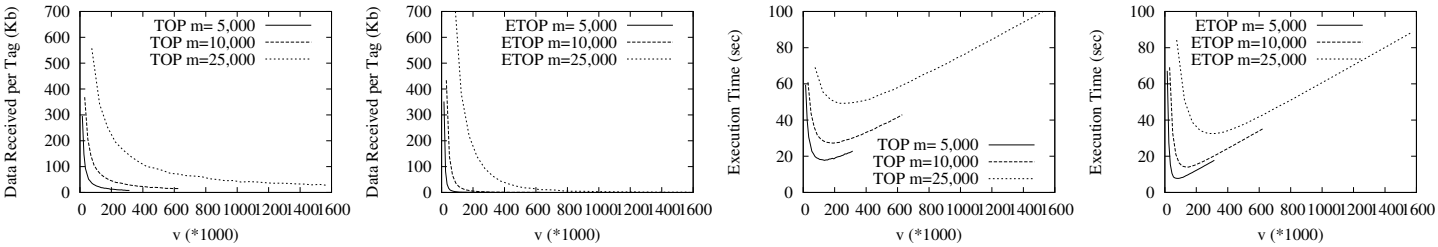

Figure 5: First plot: Energy cost of TOP with respect to v. Second Plot: Energy cost of ETOP with respect to v. Third Plot: Execution time of TOP with respect to v. Fourth Plot: Execution time of ETOP with respect to v.

[16] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," *Proc. of ACM MOBICOM*, 2006.

[17] J. Zhai and G. N. Wang, "An Anti-collision Algorithm using Two-functioned Estimation for RFID Tags," *Proc. of ICCSA*, 2005.

[18] C. Qian, H. Ngan, and Y. Liu, "Cardinality Estimation for Large-scale RFID Systems," *Proc. of IEEE PerCom*, 2008.

[19] H. Han, B. Sheng, C. C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID Tags Efficiently and Anonymously," *Proc. IEEE INFOCOM*, 2010.

[20] T. Li, S. Chen, and Y. Ling, "Identifying the Missing Tags in a Large RFID System," *Proc. of ACM Mobihoc*, 2010.

[21] L. Yang, J. Han, Y. Qi, and Y. Liu, "Identification-free Batch Authentication for RFID Tags," *Proc. of IEEE ICNP*, 2010.

[22] T. Dimitriou, "A Secure and Efficient RFID Protocol that could make Big Brother (partially) Obsolete," *Proc. of IEEE PerCom*, 2006.

[23] T. Li, S. Wu, S. Chen, and M. Yang, "Energy Efficient Algorithms for the RFID Estimation Problem," *Proc. IEEE INFOCOM*, 2010.

[24] V. Namboodiri and L. Gao, "Energy-Aware Tag Anti-collision Protocols for RFID Systems," *Proc. of IEEE PerCom*, 2007.

[25] D. Klair, K. Chin, and R. Raad, "On the Energy Consumption of Pure and Slotted Aloha based RFID Anti-collision Protocols," *Computer Communications*, 2008.

[26] S. Chen, J. Li, and M. Zhang, "Polling Protocols in RFID Systems," *Internal Technical Report, Network Information Center, University of Shanghai for Science and Technology, Shanghai, China*, 2010.

[27] W. Chen, W. Che, X. Wang, C. Huang, N. Yan, H. Min, and J. Tan, "A two-stage wake-up circuit for semi-passive RFID tag," *IEEE International Conference on ASIC*, 2009.

[28] B. Bloom, "Space / Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[29] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[30] P. Semiconductors, "I-CODE Smart Label RFID Tags," *http://www.nxp.com/documents/data_sheet/ SL092030.pdf*, Jan 2004.

[31] J. I. Capetenakis, "Tree Algorithms for Packet Broadcast Channels," *IEEE Transactions on Information Theory*, vol. 25, no. 5, 1979.