Efficient Memoryless Protocol for Tag Identification

[Extended Abstract]

Ching Law
Massachusetts Institute of Technology
ching@list.mit.edu

Kayi Lee
Massachusetts Institute of Technology
kylee@list.mit.edu

Kai-Yeung Siu
Massachusetts Institute of Technology
siu@list.mit.edu

ABSTRACT

This paper presents an efficient collision resolution protocol and its variations for the tag identification problem, where an electromagnetic reader attempts to obtain within its read range the unique ID number of each tag. The novelty of our main protocol is that each tag is *memoryless*, i.e., the current response of each tag only depends on the current query of the reader but not on the past history of the reader's queries. Moreover, the only computation required for each tag is to match its ID against the binary string in the query. Theoretical results in both time and communication complexities are derived to demonstrate the efficiency of our protocols.

1. INTRODUCTION

Recent advances in low-cost, network-aware embedded processors enable the efficient control of most devices and machines over an open communication infrastructure ranging from a small home network to the global Internet. Moreover, the emergence of low-power and low-cost sensing technologies also makes it likely that in the near future, a variety of consumer goods will be tagged with remotely readable identification tags. For example, electromagnetic tags, which are being considered to replace Uniform Product Codes (Bar Codes), can be read automatically from a distance without line-of-sight. This opens up vast new opportunities for implementing smart automated systems exhibiting intelligent, collaborative behavior that can drastically improve human productivity and efficiency. Example applications include automatic object tracking, inventory and supply chain management, and Web appliances [1].

Motivated by the need for implementing a low-cost, robust automatic identification system using electromagnetic readers and tags, we consider in this paper a tag identification problem in which a reader attempts to obtain the information (a unique identification number) stored in each individual tag within the read range. Because of the severe cost constraints in implementing these tags in practical applications[1], each tag is desired to be passive (i.e. no battery requirement) and can only have minimal built-in computing circuitry. The reader is responsible to transmit certain signals, which will power each individual tag at a distance to respond with its unique ID to the reader.

The natural question is: what protocol should the reader and the tags use so that the ID of each tag can be communicated to the reader as fast and reliably as possible? Without any coordination among the reader and the tags, the responses from the tags to the reader can collide, in which case the IDs of the tags will become illegible to the reader. This is a special case of the multiple-access communication problem and the solution to the general problem which does not require prior scheduling or central control is often referred to as the collision (or conflict) resolution protocol. This problem has been studied extensively in the past several decades. However, our applications introduce a more challenging aspect to the problem. Because of the severe computational and energy constraints in each tag, it is unreasonable to assume that the tags can communicate with each other directly and that they can maintain dynamic states of the communication process in their circuitry.

We present here a novel collision resolution protocol which allows the reader to obtain the ID from each tag within its read range, while the computational and memory requirement for each tag is minimal. In fact, the key feature of our main protocol is that each tag is *memoryless*, i.e., the current response of each tag only depends on the current query of the reader but not on the past history of the reader's queries. Moreover, the only computation required for each tag is to match its ID against the binary string in the query. The rest of the paper is organized as follows: Section 2 introduces an abstract model for the tag identification problem. Section 3 presents the basic Query-Tree (QT) protocol. Section 4 discusses related work. Section 5 analyzes the time complexity of the QT protocol. Section 6 presents a generalization of the QT protocol to handle an unreliable tag-reader communication channel. Section 7 analyzes the communi-

This work was supported in part by the MIT Auto-ID Center, with industrial sponsors consisting of the Uniform Code Council, Procter & Gamble, Gillette, International Paper, EAN International and CHEP. Please forward any future correspondence to Kai-Yeung Siu at siu@list.mit.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DIAL M Workshop 2000 Boston MA USA
Copyright ACM 2000 1-58113-301-4/00/08...$5.00

[1]For example, to replace the Bar Codes with electromagnetic tags, the objective is to bring the manufacturing cost of each tag to less than 1 cent.

cation complexity of the QT protocol and introduces two variants for reducing the communication complexity. We note that for brevity we omit some of the proofs in Sections 5 and 7. Detailed proofs can be found in the full version of this paper [7]. Concluding remarks are given in Section 8.

2. TAG IDENTIFICATION

A tag identification system consists of one reader and n tags. The reader is a powerful entity with abundant memory and computation power. On the other hand, the tags are limited in memory and computation power.

There is a single communication channel between the reader and the tags. However, the tags are not able to exchange messages among each other. The reader can broadcast messages to the tags. Upon receiving a message, each tag can optionally send a response back to the reader. If only one tag responds, the reader receives the message. But if more than one tag respond, their messages would collide on the communication channel, and thus cannot be received by the reader. In this case, the reader detects a collision on the channel but nothing else.

Each tag $i \in \{1, \ldots, n\}$ has a unique ID string in $\{0,1\}^k$, where k is the length of the ID string. At the beginning, the reader does not know anything about the tags. A tag identification protocol specifies the algorithms for the reader and the tags, so that the reader can collect all the tag IDs.

3. THE QUERY-TREE PROTOCOL

In this section, we will introduce the *Query Tree* (QT) protocol. Our time and communication complexity analyses will be based on this protocol. In Section 5.4 we will introduce different variations of the protocol to optimize the running time. In Section 7 we will introduce the QT-sl and QT-im variants for optimizing the communication complexity.

The QT algorithm consists of rounds of queries and responses. In each round, the reader asks the tags whether any of their IDs contains a certain prefix. If more than one tag answer, then the reader knows that there are at least two tags having that prefix. The reader then appends symbol 0 or 1 to the prefix, and continues to query for longer prefixes. When a prefix matches a tag uniquely, that tag can be identified. Therefore, by extending the prefixes until only one tag's ID matches, the algorithm can discover all the tags. The following describes the protocol:

The QT Protocol

Let $\mathcal{A} = \cup_{i=0}^{k} \{0,1\}^i$ be the set of binary strings with length at most k. The state of the reader is a pair (Q, M), where

1. queue Q is a sequence of strings in $\mathcal{A}$;
2. memory M is a set of strings in $\mathcal{A}$.

A query from the reader is a string q in $\mathcal{A}$.

A reply from a tag is a string w in $\{0,1\}^k$.

Reader For convenience, let us define the queue Q be $\langle \varepsilon \rangle$, where ε is the empty string, and memory M be empty initially.

ID: {000, 001, 101, 110}

Step	Query	Response
1	ε	collision
2	0	collision
3	1	collision
4	00	collision
5	01	no response
6	10	101
7	11	110
8	000	000
9	001	001

Figure 1: Communication between the reader and the tags with the QT Protocol.

1. Let $Q = \langle q_1, q_2, \ldots, q_l \rangle$.
2. Broadcast query q_1 to the tags.
3. Update Q to be $\langle q_2, \ldots, q_l \rangle$.
4. On receiving the responses from the tags:
   - If the reply is string w, then insert string w into memory M.
   - If a collision is detected at communication channel, then set Q to be $\langle q_2, \ldots, q_l, q_1 0, q_1 1 \rangle$.
   - If there is no reply, do nothing.

Repeat the above procedure until Q is empty.

Tag Let $w = w_1 w_2 \cdots w_k$ be the tag's ID. Let w be the query string received from the reader. If $q = \varepsilon$ or $q = w_1 w_2 \cdots w_{|q|}$, then the tag sends string w to the reader.

We note that when more than one tag try to respond at the same time, the reader will detect a collision instead of receiving the messages. An example of the communication between the reader and the tags in the protocol is illustrated in Figure 1.

In reality, it is not possible for the reader to send the empty string ε. Thus, in practice, the protocol should start with strings 0 and 1. This will only improve the performance unless there is only one tag, since otherwise the empty string will guarantee a conflict.

4. RELATED WORK

Before presenting the analysis of the QT protocol, we first discuss related previous work.

Our QT protocol is closely related to conflict resolution algorithms in the area of multiple access communication [6, 12, 2]. Since our QT protocol is designed for tag identification, which is quite different from those applications considered in prior research on conflict resolution protocols, the specific mechanism of our protocol is also quite different. However, the underlying algorithm shares similar insights with previous work. Conflict resolution algorithms were introduced by Capetanakis [3], and Tsybakov and Mikhailov [12]. It was further studied by Massey [8], Fayolle *et al.* [5], and Mathys

and Flajolet [9]. In particular, [9] contains a detailed theoretical treatment. For a comprehensive survey and a bibliography on conflict resolution algorithms, see Molle and Polyzos [10].

Since conflict resolution algorithms have already been widely studied, our discussion will focus on the properties of the QT protocol as applied to the problem of tag identification. In particular, we note the following differences between the tag identification problem and the general multiple access communication problem:

- The tags are very limited in computational power and memory, whereas computation and memory are not the limiting constraints in multiple access communication.
- The number of tags in our problem is fixed during the run of the algorithm, whereas the general multiple access problem usually models network traffic as a stochastic process.
- Communication bandwidth is much more constrained in the tag identification problem than in the general multiple access problem. Thus it is highly desirable to reduce the number of bits transmitted in the tag identification problem. This will be further discussed in Section 7.
- When fault tolerance issues are considered, the fault model in the tag identification problem is also different from the general multiple access problem. We will discuss the issue of fault tolerance in Section 6.

We also want to emphasize the following characteristics of the QT protocol:

- The tags are memoryless: a tag's response depends on the current query and its tag ID only.
- The QT protocol is deterministic and does not assume any randomization technique.
- The only computation required for a tag is prefix-matching its own predetermined ID.

Chan *et al.* [4] suggests a direct implementation of a conflict resolution algorithm commonly studied in previous work for the tag identification problem. However, we note that this approach suffers from the following shortcomings.

- A $\log_2 k$-bit register is needed to store certain state information in each tag.
- Each tag needs to be able to increment and decrement its counter.
- Each tag needs to generate a random bit at each step.
- The number of bits sent by each tag is large compared to our QT-sl protocol to be introduced in Section 7. In fact, the algorithm in [4] is optimized for the number of messages sent by the reader, while our Query-Tree algorithm is optimized for implementation simplicity and power consumption.

5. TIME COMPLEXITY

In this section, we analyze the time complexity of the QT protocol. Assuming that each query-response step takes a fixed amount of time, we count the number of queries sent by the reader in a complete execution of the protocol. We define the *identification time* of the QT protocol, denoted by T_S, as the number of queries sent by the reader in order to identify a set S of tags. As we have discussed in the preceding section, the underlying algorithm of QT is similar to the conflict resolution algorithms studied in some previous work. Using similar analysis from [8], we can show that for $n = |S| \geq 4$,

$$2.881n - 1 \leq \mathrm{E}[T_S] \leq 2.887n - 1 \qquad (1)$$

for a uniformly distributed random set S, where $\mathrm{E}[T_S]$ is the expected identification time. This gives us the average time complexity of the QT protocol. In Section 5.2, we discuss the worst-case time complexity of the protocol. We show that in the worst case, it takes $n \cdot (k + 2 - \log n)$ steps to identify all the n tags. In Section 5.3, we argue that with high probability, the running time of the protocol is $O(n)$.

To help our analysis in the current section and subsequent sections, we introduce the notion of a *query tree*, which describes the complete dialogue between the reader and the tags in an execution of the QT protocol. Knowing the size of the query tree, we can find out the identification time of the QT protocol.

5.1 Query Tree

A query tree is a full binary tree (a binary tree in which each node has either two children or no children) capturing the complete reader-tags dialogue of the QT protocol. For a given execution of the protocol, there is a one-to-one correspondence between every node in the query tree and every query sent by the reader. Therefore, the *size*, i.e. number of nodes, in the query tree is equal to the number of queries sent by the reader.

For any node x in the query tree, let $l(x)$ and $r(x)$ be the left child and right child of x respectively. If x is a leaf node, then $l(x)$ and $r(x)$ are defined to be NIL.

Definition of a Query Tree

Suppose in an execution of the QT protocol, the reader has sent the set Q of query strings. The query tree of the QT protocol execution is defined recursively by Q.

1. The root of the query tree corresponds to the query string ε.
2. If the query tree node x corresponds to the query string q, and both $q0, q1 \in Q$, then $l(x)$ and $r(x)$ are query tree nodes that correspond to the query strings $q0$ and $q1$ respectively. Otherwise, both $l(x)$ and $r(x)$ equal NIL.

The above definition implies that an internal node of a query tree corresponds to a query string that results in a collision in the communication channel. On the other hand, a leaf corresponds to a query string that results in either no reply or a response from exactly one tag. To facilitate our discussion, we shall call a leaf *white* if the corresponding query

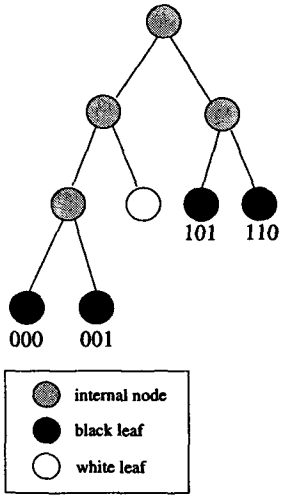


Figure 2: The query tree for the example in Figure 1

string results in no reply, and *black* otherwise. Figure 2 shows the query tree for the example in Figure 1.

Structure of a Query Tree

From the definition of query tree, we have the following observations:

1. The height of a query tree is at most k, since the query string sent out by the reader is at most k bits long.
2. If x is an internal node, then x has at least two black leaf descendants. This follows from the fact that each black leaf corresponds to a unique tag ID and each internal node corresponds to a query that results in a collision.

5.2 Worst-Case Time Complexity

The number of queries sent by the reader equals the size of the query tree. Given a set S of n tags, let Y_S be the query tree for S. Also, let R_S be number of internal nodes in the tree Y_S. Since a query tree is a full binary tree, the size of the tree is simply $2R_S + 1$.

A simple argument can give a bound on the size of Y_S. In a query tree, any internal node is an ancestor of some black leaf. For each black leaf, it has at most k ancestors, which are internal nodes. This gives us $R_S \leq kn$. Therefore, it follows that the size of the tree equals $2R_S + 1$, which is no more than $2nk + 1$.

An improved result is stated in the following:

THEOREM 1. *The number of queries sent by the reader to identify n tags is at most $n \cdot (k + 2 - \log n)$.*

PROOF. The number of queries sent by the reader to identify n tags equals the size of the corresponding query tree, which has exactly n black leaves. In Appendix A, it is shown that the size of any query tree with exactly n black leaves is no more than $n \cdot (k + 2 - \log n)$. □

5.3 Probabilistic Analysis

Here we show that with high probability, the running time of the QT protocol is $O(n)$.

Mathys and Flajolet [9] claimed that the variance of the running time can be shown to be linear in n, as $n \to \infty$. This would be sufficient to show that the running time is linear in n with high probability. However, the derivation was omitted in [9] because it is "rather lengthy and complicated".

In the following we will present a proof that QT has linear running time with high probability. We note that by bounding the number of white leaves (as introduced in Section 5), we essentially bound the total size of the tree.

Let W_n be the random variable of the number of white leaves in a query tree with n black leaves. We will apply the Chernoff bound on the upper tail of the distribution of W_n.

We first need the following technical lemma.

LEMMA 1.

$$\Pr\{W_n \geq a\} \leq e^{0.4(n-a)}.$$

PROOF. For $n \geq 2$, we have the following recurrence:

$$\mathrm{E}\left[e^{0.4W_n}\right] = \sum_{i=0}^{n} P_{i,n-i} \mathrm{E}\left[e^{0.4(W_i + W_{n-i})}\right],$$

where W_i and W_{n-i} are the number of white leaves in the two subtrees. Since W_i and W_{n-i} are independent, we can write Equation (5.3) as

$$\mathrm{E}\left[e^{0.4W_n}\right] = \sum_{i=0}^{n} P_{i,n-i} \mathrm{E}\left[e^{0.4W_i}\right] \mathrm{E}\left[e^{0.4W_{n-i}}\right].$$

Then we can prove by induction that for $n \geq 2$,

$$\mathrm{E}\left[e^{0.4W_n}\right] \leq e^{0.4n}. \tag{2}$$

Applying the Chernoff Bounds[11, p.39] to Equation (2), Lemma 1 follows. □

We now show that the running time of QT is $O(n)$ with high probability. In particular, the probability that QT takes at least cn steps decreases exponentially with size n.

THEOREM 2. *The probability the QT protocol takes at least cn steps to identify n tags is at most $e^{-0.4n(c/2-2)}$.*

PROOF. When the size of the query tree is larger than $cn - 1$, the number of white leaves is at least $cn/2 - n$. By Lemma 1,

$$\Pr\{W_n \geq a\} \leq e^{-0.4(a-n)}$$

for all $a > 0$. Let $a = cn/2 - n$, then,

$$\begin{aligned}\Pr\{W_n \geq cn/2 - n\} &\leq e^{-0.4((cn/2-n)-n)} \\ &= e^{-0.4n(c/2-2)}.\end{aligned}$$ □

5.4 Further Enhancement

Here we discuss several techniques of reducing the running time of the QT protocol.

5.4.1 *Shortcutting*

Consider any internal node of the query tree. This corresponds to a collision for certain prefix q during an execution of the QT protocol. The algorithm will continue to search for the tags by appending 1 and 0 to the prefix q. Without loss of generality, assume that the algorithm chooses to search for 0 first. If it turns out that there are no tags with prefix $q0$, then we know that there are at least two tags with prefix $q1$. Therefore, the reader should as well skip the prefix $q1$.

In the shortcutting QT algorithm, the reader can randomly choose the order of sending $q1$ and $q0$. But this would be unnecessary if we assume that the tag IDs are uniformly distributed.

This technique is similar to the modified conflict resolution algorithm in the literature [8, 9], and has been shown to give an improved expected running time bound of $2.665n - 1$.

5.4.2 *Aggressive Advancement*

Assume the reader knows that there are at least n unrecognized tags with prefix q. For example, this could be an a prior knowledge: maximum number of items in a checkout counter. Or the reader can detect the strength of the response from the tags to estimate the number of tags. When n is large, it is very likely that the responses for $q1$ and $q0$ will collide. The probability that either one of the queries does not result in a collision is $2 \cdot (\frac{1}{2^n} + n \cdot \frac{1}{2^n}) = \frac{n+1}{2^{n-1}}$. Now suppose we extend the prefix string by two bits. That is, the reader will query $q00$, $q01$, $q10$, and $q11$. In this case, we save two queries $q1$ and $q0$ with probability $1 - \frac{n+1}{2^{n-1}}$. Note that we send more queries (campared with the original QT protocol) only in the case where both $q0$ and $q1$ have exactly one tag with a matching prefix. This cannot happen when $n \geq 3$.

This technique is equivalent to the "Q-ary tree conflict resolution" as analyzed by [9], which showed that a *3-ary* tree is optimal without shortcutting, but the basic *2-ary* tree is preferred if shortcutting is used.

5.4.3 *Categorization*

If the reader has some information about the types of the tags, then it is possible to speed up the protocol. For example, suppose a set S of IDs is given. Assume the reader knows that set S can be partitioned into $S_1, \ldots, S_m$ such that all IDs in S_i have prefix q_i. Now the reader can just identify each set S_i independently. In particular, if we can partition the tags into m groups, then the upper bound on the expected running time is improved to $2.887n - m$.

6. FAULT TOLERANCE

We now introduce a dynamic version of the QT protocol that is useful when the communication between the tags and the reader is unreliable. We assume that there is a probability p that a given tag would fail to reply to a query. We assume that these failures are statistically independent events. During any execution of the QT protocols, some tags may not be identified due to these failures. We will develop a QT algorithm with multiple tries such that the probability of missing any tags is low.

It is impossible to identify all tags with 100% certainty if we want our protocol to terminate. Instead, our algorithm is designed to trade off between running time and the probability of missing some tags. In particular, in the QT^l protocol, if the reader receives no collision on a query string q, it will repeat sending the the same query string until it detects a collision or the same query has been sent for l times. The reader will summarize the responses to all such queries and determine whether there are multiple, one or no tags having the prefix q. Specifically, while the reader is repeatedly sending the query string, it will summarize the responses as follows:

- If the reader detects a collision for the query string, it will conclude that more than one tag have a matching prefix.
- If in the course of the l queries, no tag has replied, the reader will conclude that no tag has a matching prefix.
- If in the course of the l queries, it receives response(s) from exactly one tag, the reader will conclude that only the tag has a matching prefix.
- If in the course of the l queries, it receives responses from different tags, the reader will conclude that more than one tag have a matching prefix. It will then behave as if a collision is detected.

Since the same query string is sent for multiple times, it is unlikely that a tag with a matching prefix is not identified.

THEOREM 3. *With failure probability $p \leq 1/2$, the expected running time of the QT^l protocol is at most $\frac{3l+9}{2}n - 1$.*

PROOF. There is a probability of p^n that all tags fail to respond. And for $n > 1$, there is a probability of $np^{n-1}(1-p)$ that exactly one tag responses.

Let $T(n)$ be the expected running time of the QT^l algorithm when there are n tags to be identified.

First, for the base cases,

$$T(0) = l$$
$$T(1) = l$$

For $n \geq 2$, let

$$p_n = p^n + np^{n-1}(1-p)$$

$$T(n) \leq p_n T(n) + (1-p_n)\sum_{i=0}^{n} P_{i,n-i} \cdot (T(i) + T(n-i)) + 1 \quad (3)$$

We can rewrite Equation (3) as

$$(1-p_n)T(n) = (1-p_n)\sum_{i=0}^{n} P_{i,n-i} \cdot (T(i) + T(n-i)) + 1$$

thus,

$$T(n) = \sum_{i=0}^{n} P_{i,n-i} \cdot (T(i) + T(n-i)) + \frac{1}{1-p_n} \quad (4)$$

Then we can prove by induction that

$$T(n) \leq \frac{3l+9}{2} n - 1. \quad \square$$

THEOREM 4. *With failure probability p, the probability that the QT^l protocol does not recognize a particular tag is at most p^l.*

PROOF. Consider an arbitrary tag w. For the tag w to be unidentified, the reader must have repeated sending some prefix of w for l times. In addition, the tag w must have failed to respond to all these l queries. The probability for this to happen is at most p^l. $\square$

COROLLARY 1. *With failure probability p, the probability that the $\mathsf{QT}^{(c+1)\log_{1/p} n}$ protocol does not identify all tags is at most $1/n^c$.*

PROOF. The probability that a certain tag is not identified is at most $p^{(c+1)\log_{1/p} n}$. Therefore, the probability that any tag is unidentified is at most $np^{(c+1)\log_{1/p} n} = 1/n^c$. $\square$

7. COMMUNICATION COMPLEXITY

In this section we turn our attention to the communication complexity of the protocol. The *reader communication complexity* is the number of bits sent by the reader; and the *tag communication complexity* is the number of bits sent by a tag. The tag communication complexity is especially important because it is desirable to minimize the power consumption of the tags.

We will first derive the communication complexities of our QT protocol and then introduce several variants that improve upon the performance of QT.

7.1 Basic Query-Tree

In the followings, we will first find the expected number of collisions experienced by a tag. We assume that the bit length k of each tag ID is infinite. This will give us an upper bound for cases where k is finite. We show that in the QT protocol, the expected number of responses a tag makes is no more than $2.21 \log_2 n + 3.19$, where n is the total number of tags.

In the algorithm QT, each tag responds to query strings that match its prefix. It will experience a collision only if there is some other tag having the same prefix, which is the query string sent by the reader.

Let w be the ID of an arbitrary tag in a set of n tags, in which the IDs are uniformly distributed. Let C_w be the number of collisions the tag experiences during the execution of the QT protocol. In addition, let $I_w^j, j = 0, 1, 2, \ldots$, be an indicator variable such that:

$$I_w^j = \begin{cases} 0 & \text{if none of any other } n-1 \text{ tags has the same} \\ & j\text{-bit prefix as } w, \\ 1 & \text{otherwise.} \end{cases}$$

Then we have the following equation:

$$C_w = \sum_{j=0}^{\infty} I_w^j.$$

By linearity of expectation,

$$\mathrm{E}[C_w] = \sum_{j=0}^{\infty} \mathrm{E}\left[I_w^j\right].$$

Let w_j be the j-bit prefix of w. Then for each $j = 0, 1, 2, \ldots$,

$$\begin{aligned} \mathrm{E}\left[I_w^j\right] &= \Pr\{\text{some other tag ID has prefix } w_j\} \\ &= 1 - \Pr\{\text{all other IDs do not have prefix } w_j\} \\ &= 1 - (\Pr\{\text{an ID does not have prefix } w_j\})^{n-1} \\ &= 1 - (1 - 2^{-j})^{n-1}. \end{aligned}$$

Therefore, the expected number of conflicting responses the tag experiences is given by:

$$\begin{aligned} \mathrm{E}[C_w] &= \sum_{j=0}^{\infty} \mathrm{E}\left[I_w^j\right] \\ &= \sum_{j=0}^{\infty} \left(1 - (1 - 2^{-j})^{n-1}\right). \end{aligned} \quad (5)$$

A bound on $\mathrm{E}[C_w]$ is derived in Theorem 5, which depends on the following technical lemma.

LEMMA 2. *For all $n \geq 2$,*

$$\sum_{j=0}^{\infty} 2^{-j}(1 - 2^{-j})^n < \frac{\log_2 e + 2e^{-\frac{2}{3}} + e^{-\frac{1}{3}}}{n+1}.$$

PROOF. Omitted for brevity. $\square$

Now we are ready to state the theorem and prove it.

THEOREM 5. *For a system with n tags, a tag is expected to experience no more than $2.21 \log_2 n + 3.19$ conflicts before it successfully transmits its ID.*

PROOF. From Equation (5), the expected number of conflicting responses a tag experiences, $\mathrm{E}[C_w]$, is given by:

$$\mathrm{E}[C_w] = \sum_{j=0}^{\infty} 1 - (1 - 2^{-j})^{n-1}.$$

We can prove by induction that

$$\sum_{j=0}^{\infty} 1-(1-2^{-j})^{n-1} \le \sum_{j=1}^{n-1} \frac{C}{j},$$

where $C = \log_2 e + 2e^{-\frac{2}{3}} + e^{-\frac{1}{3}} \approx 3.19$.

This implies

$$\begin{aligned} \mathrm{E}[C_w] &\le (1+\ln(n-1)) \cdot C \\ &< 2.21 \log_2 n + 3.19. \quad \square \end{aligned}$$

THEOREM 6. *Let there be n tags to be identified. The expected reader communication complexity for* QT *is at most* $2.89kn$. *The expected tag communication complexity is at most* $2.21k \log_2 n + 4.19k$.

PROOF. Since the expected running time for the QT protocol is at most $2.887n - 1$, and the length of each query is at most k. Therefore the expected total number of bits sent by the reader is at most $2.89kn$.

Theorem 5 implies that each tag is expected to respond for $3.19 + 2.21 \log_2 n + 1 = 4.19 + 2.21 \log_2 n$ times. On each step, the tag sends a k-bit ID. Therefore, the expected tag communication complexity is at most

$$2.21k \log_2 n + 4.19k. \quad \square$$

7.2 Short-Long Queries

Now we introduce the QT-sl (Query-Tree short-long) protocol that reduces the number of bits transmitted. We note that in QT, a lot of the k-bit responses from the tags would end up in collisions. To minimize these wastes, we can have two types of queries from the reader. The short queries will only induce 1-bit responses from the tags, while the long queries will induce the full tag IDs. The reader will send a long query only when it knows that only one tag matches the prefix.

The QT-sl Protocol

Let $\mathcal{A} = \cup_{i=0}^{k} \{0,1\}^i$ be the set of binary strings with length at most k. The state of the reader is a pair (Q, M), where

1. queue Q is a sequence of strings in $\mathcal{A}$;
2. memory M is a set of strings in $\mathcal{A}$.

A query from the reader is a pair (c, w), where $c \in \{\texttt{short}, \texttt{long}\}$ and $w \in \mathcal{A}$.

A reply from a tag is a string 1 or a string in $\{0,1\}^k$.

Reader For convenience, let us define the queue Q be $\langle \varepsilon \rangle$, where ε is the empty string, and memory M be empty initially.

1. Let $Q = \langle q_1, q_2, \ldots, q_l \rangle$.
2. Broadcast query $(\texttt{short}, q_1)$ to the tags.
3. Update Q to be $\langle q_2, \ldots, q_l \rangle$.
4. On receiving the responses from the tags:
   - If the reply is 1, then
     (a) broadcast query $(\texttt{long}, q_1)$ to the tags;
     (b) insert the resulting response string w into memory M.
   - If a collision is detected at the communication channel, then set Q to be $\langle q_2, \ldots, q_l, q_1 0, q_1 1 \rangle$.
   - If there is no reply, do nothing.

Repeat the above procedure until Q is empty.

Tags Let $w = w_1 w_2 \cdots w_k$ be the tag's ID. Let (c, q) be the query received from the reader. If $q = \varepsilon$ or $q = w_1 w_2 \cdots w_{|q|}$, then

- if command c is `short`, send string 1 to the reader;
- if command c is `long`, send string w to the reader.

THEOREM 7. *Let there be n tags to be identified. The expected reader communication complexity of* QT-sl *is at most* $3.89kn + 3.89n$. *The expected tag communication complexity of* QT-sl *is at most* $2.21 \log_2 n + k + 4.19$.

PROOF. Note that with QT-sl protocol, we need one extra bit to specify whether the query is `short` or `long`.

The expected total number of `short` and `long` queries is at most $3.887n - 1$. Each query is at most $k+1$-bit long, thus the expected reader communication complexity is at most

$$(3.887n - 1)(k+1) < 3.89kn + 3.89n.$$

Theorem 5 implies that each tag is expected to respond for $4.19 + 2.21 \log_2 n$ times. For each short query, the tag sends a 1 response. For the long query, the tag sends a k-bit ID. Therefore, the expected tag communication complexity is at most

$$2.21 \log_2 n + k + 4.19. \quad \square$$

7.3 Incremental Matching

Here we introduce another technique to further reduce the expected reader communication complexity when $\log_2 n$ is small compared to k. However, this optimization requires a tag to remember the bit position of the prefix it has matched so far. Therefore, the modified protocol is no longer memoryless.

The algorithm QT-im (Query-Tree incremental-matching) is very similar to QT-sl. Thus we will only describe the difference between these two protocols.

First, the QT-im protocol will follow the query tree in a preorder fashion.

In QT-im, each tag has a bit marker $b \in \{1, \ldots, k\}$. A tag is *active* if it has responded 1 in the previous step. When the tag is active, upon receiving the a query, the tag matches

	Reader	Tag	Total
QT	$2.89kn$	$2.21k\log_2 n + 4.19k$	$2.21kn\log_2 n + 7.08kn$
QT-sl	$3.89kn + 3.89n$	$2.21\log_2 n + k + 4.19$	$2.21n\log_2 n + (8.08 + 4.89k)n$
QT-im	$4.42n\log_2 n + 12.18n$	$2.21\log_2 n + k + 4.19$	$6.63n\log_2 n + (16.37 + k)n$

Table 1: Summary of communication complexities of QT, QT-sl, and QT-im. We note that $\log_2 n \leq k$ and k is around 96 in practice.

the query string starting from bit b. If the matching is successful, then bit marker b is incremented by 1. Any active tag that mismatches would go into the *transient* state. A transient tag will become *inactive* in the next query unless that query contains the `reactivate` command.

When a `long` query is received, all tags would reset the bit marker to 1 and become active again. The active tag, upon receiving a `long` query, will also respond with its full tag ID.

Whenever a `short` query does not receive any response the reader will send a `reactivate` query, which is equivalent to a `short` query except that all transient tags will become active again. Each `short` command takes 1 bit as before, but the `reactivate` and `long` commands would need 2 bits each.

With these extra tag functionalities, the reader can then send the prefixes incrementally. For example, if the reader sent q in the previous step and the reader plans to send $q0$ or $q1$, it can simply send 0 or 1 instead. Moreover, it is no longer necessary to supply a prefix with the long query.

The tag communication complexity of QT-im is the same as that of QT-sl. However, the number of bits sent by the reader is reduced.

THEOREM 8. *The expected reader communication complexity of* QT-im *protocol is at most* $4.42n\log_2 n + 12.18n$.

PROOF. We can partition the queries in groups such that each group ends with a long query. Since exactly one tag is identified for each group of queries, there are n groups in total. We can find the number of bits transmitted in each group. Theorem 5 implies that on average there are at most $2.21\log_2 n + 3.19 + 1 = 2.21\log_2 n + 4.19$ short queries in a group that result in either a collision or a single response. Since each short query is 2 bits long. The total number of bits transmitted for short queries that result in a collision or a single response, over all the n groups, is:

$$4.42n\log_2 n + 8.38n.$$

Each long query is 2 bits long. Therefore, in total the reader sends $2n$ bits for long queries. For each white leaf, the reader sends a 2-bit short query that corresponds to the white leaf, and another 2 bits for the `reactivate` command. Therefore, 4 bits are sent for each white leaf discovered. Equation (1) implies that on average there are at most $0.444n$ white leaves. Therefore, the expected `reactivate` overhead is at most $1.8n$. In summary, the expected reader communication complexity is at most

$$\begin{aligned} & 4.42n\log_2 n + 8.38n + 2n + 1.8n \\ = \; & 4.42n\log_2 n + 12.18n. \quad \square \end{aligned}$$

7.4 Summary and Comments

Table 1 summarizes the communication complexities of QT, QT-sl, and QT-im, together with simple lower bounds. We showed that we can reduce the communication complexities with the more complicated implementations. Lastly we note that it is possible to achieve $O(n)$ reader communication complexity by implementing reader commands to decrease the bit marker of the tags. Such algorithm (like the one suggested in [4]) would resemble the implementation of the conflict resolution algorithms in computer networking more closely. However, we have decided not to discuss it in detail here because the implementation would be too complicated for the tags in practice.

8. CONCLUDING REMARKS

In this paper we introduce Query-Tree (QT) as the first memoryless protocol for the tag identification problem. The Query-Tree protocol has simple and deterministic implementation for the tags.

We show that QT runs in $n(k+2-\log_2 n)$ steps in the worst case, but in $O(n)$ time with high probability. In addition, we show that QT can be extended to find all tags with high probability, even if each tag can fail independently during each step. We also study the communication complexity of the QT protocol, and suggest two variations for further reducing the complexity. The communication complexity results are summarized in Table 1.

We note that given multiple electromagnetic communication channels, the QT protocol can be parallelized. We are currently investigating how the performance of a parallel QT scales with the number of channels. The basic approach is to statically assign the channels to different tag ID prefixes, so that the reader will identify the tags using the preassigned channel. This approach essentially partitions the set of tags according to their prefixes, so that each group of tags will be identified independently in parallel.

A more sophisticated approach is to dynamically assign the channels. In this approach, whenever a channel is idle, it will be reused by the reader to communicate with the currently unidentified tags. Therefore, the channels can be utilized more efficiently.

9. REFERENCES

[1] MIT Auto-ID Center. `http://auto-id.mit.edu/`.

[2] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice-Hall, second edition, 1992.

[3] J. I. Capetanakis. *The Multiple Access Broadcast Channel: Protocol and Capacity Considerations.* PhD thesis, 1977.

[4] Shun Chan, Harley Heinrich, Dilip Kandlur, and Arvind Krishna. U.S. Patent: Multiple item radio frequency tag identification protocol. Patent number: US5550547. August 1996.

[5] G. Fayolle, P. Flajolet, M. Hofri, and P. Jacquet. Analysis of a stack algorithm for random access communication. *IEEE Transactions on Information Theory*, IT-31(2):244–254, March 1985. (Special Issue on Random Access Communication, J. Massey editor).

[6] Robert G. Gallager. A perspective on multiaccess channels. *IEEE Transactions on Information Theory*, IT-31(2):124–142, March 1985. (Special Issue on Random Access Communication, J. Massey editor).

[7] C. Law, K. Lee, and K. Siu. Efficient memoryless protocol for tag identification (full version paper). `http://perth.mit.edu/pub/DIALM00.ps`.

[8] J. L. Massey. Collision-resolution algorithms and random-access communications. *Multi-User Communication Systems*, pages 73–99, 1981.

[9] P. Mathys and P. Flajolet. Q-ary collision resolution algorithms in random access systems with free or blocked channel access. *IEEE Transactions on Information Theory*, IT-31(2):217–243, March 1985. (Invited Paper, Special Issue on Random Access Communication, J. Massey editor).

[10] Mart L. Molle and George C. Polyzos. Conflict resolution algorithms and their performance analysis. Technical report, 1993.

[11] Sheldon Ross. *Stochastic Processes.* John Wiley & Sons, Inc., second edition, 1996.

[12] Boris Tsybakov. Survey of USSR contributions to random multiple-access communications. *IEEE Transactions on Information Theory*, IT-31(2):143–165, March 1985. (Special Issue on Random Access Communication, J. Massey editor).

APPENDIX

A. BOUNDING THE QUERY TREE SIZE

Our objective in this section is to give an upper bound on the size of a query tree with n black leaves. Let T be a query tree with n black leaves. Since it is a full binary tree, the number of nodes in T is simply $2l-1$, where l is the number of leaves in T. Therefore, our goal in this section is to bound the number of leaves l in T. The result will be stated in Theorem 9, which depends on the following Lemmas.

LEMMA 3. *For any query tree with height k and two black leaves, the number of leaves in the tree is at most $k+1$.*

PROOF. Suppose there are $m \geq k+2$ leaves in the query tree. Then the tree has at least $k+1$ internal nodes. Since the height of the query tree is at most k, there exist two internal nodes in the tree whose depth is the same. Therefore, these two nodes do not have any common descendants. As a result, one of them must have fewer than two black leaf descendants, since there are totally two black leaves in the query tree. This contradicts that every internal node must have at least two black leaf descendants. □

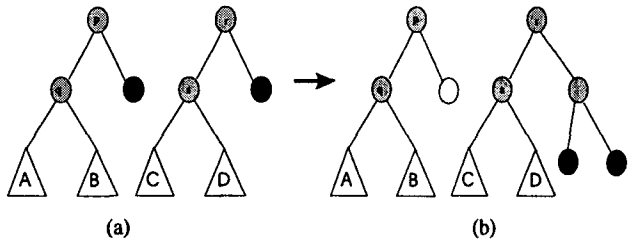


Figure 3: (a): Two subtrees of a query tree with an unpaired black leaf. (b): The modified subtrees. Note that there is an extra white leaf and internal node in the modified query tree.

LEMMA 4. *Suppose T is the largest query tree with exactly n black leaves. If n is even, then the sibling of any black leaf in T is also a black leaf. In n is odd, the same is true except for one black leaf, whose sibling is an internal node.*

PROOF. First note that the sibling of a black leaf cannot be a white leaf, since otherwise the parent of the black leaf will have only one black leaf descendant. Now suppose there are two black leaves in T whose siblings are internal nodes. Then Figure 3 illustrates how we can construct a new query tree that is larger than T, which contradicts that T is the largest query tree. □

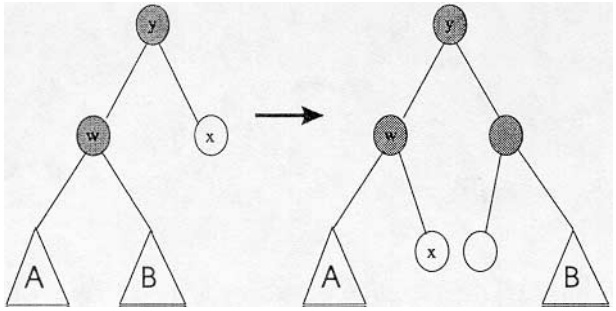
LEMMA 5. *If T is the largest query tree with exactly n black leaves, where n is odd, then there exists a query tree T' that has exactly $n-1$ black leaves and has the same size as T.*

PROOF. By Lemma 4 there is a black leaf in T whose sibling is an internal node. By replacing the black leaf by a white leaf, the modified tree T' is still a valid query tree. In addition, it has $n-1$ black leaves and has the same size as T. □

Because of Lemma 5, we only consider the case where n is even. Suppose T is the largest query tree with exactly n black leaves. By Lemma 4, we can pair up all the sibling black leaves (b_i^1, b_i^2) in T.

To count the number of leaves in T, we first "cut away" subtrees from T to form a set of subtrees $\mathcal{Q}$, so that any leaf in T belongs to some subtree in $\mathcal{Q}$, as stated in Lemma 6. As a result, the number of leaves in T is at most the total number of leaves in the subtrees in $\mathcal{Q}$. The set $\mathcal{Q}$ is defined as follows:

$$\mathcal{Q} = \{S_i | S_i \text{ is the largest subtree of } T \text{ that contains only } (b_i^1, b_i^2) \text{ as its black leaf descendants}\}. \quad (6)$$



(a) (b)

Figure 4: Modifying a query tree with the structure in (a) into a larger query tree in (b). The tree in (b) has one more white leaf and internal node than (a).

LEMMA 6. *Suppose T is the largest query tree with exactly n black leaves, where n is even and positive, then any leaf in T will appear in some subtree in $\mathcal{Q}$.*

PROOF. By definition of $\mathcal{Q}$, every black leaf must appear in some subtree in $\mathcal{Q}$. Suppose there is a white leaf x that does not appear in any subtree in $\mathcal{Q}$. Let y denote the parent of x and S denote the subtree rooted at y. Then at least two pairs of black leaves must appear in S. Suppose only one pair of black leaves appear in S. The fact that $S \notin \mathcal{Q}$ implies there is a different subtree $S_i \in \mathcal{Q}$ of T such that S_i contains the same pair of black leaves and it is larger than S. This implies S is a subtree of S_i. Since $S_i \in \mathcal{Q}$, the white leaf x does not appear in S_i. As a result, the fact that x appears in S contradicts that S is a subtree of S_i.

Given that at least two pairs of black leaves appear in S, Figure 4 shows the structure of the subtree S. The figure illustrates how we can modify S to construct a new query tree S' that has one more white leaf than S. If we replace S by S' in the query tree T, it would give a new query tree that is larger than T. This contradicts that T is the largest query tree among all the trees with the same number of black leaves. □

As a result, we can count of total number of leaves in $\mathcal{Q}$ to give an upper bound of the number of leaves in T. Since every subtree in $\mathcal{Q}$ has exactly two black leaves, we can apply Lemma 3 to count the number of leaves in each subtree.

Now let $\mathcal{Q}$ be a set of subtrees constructed according to (6). For each subtree $S_i \in \mathcal{Q}$, let $root(S_i)$ denote the root of the tree and let $depth(S_i)$ denote the depth of the node $root(S_i)$ in T.

LEMMA 7. $-\sum_{S_i \in \mathcal{Q}} depth(S_i) \leq -\frac{n}{2}\log\frac{n}{2}$.

PROOF. The node $root(S_i)$ has depth $depth(S_i)$ in the original tree T. Since T is a binary tree, and all the trees in $\mathcal{Q}$ are disjoint, if we define $h(S_i) = 2^{-depth(S_i)}$, we have:

$$\sum_{S_i \in \mathcal{Q}} h(S_i) \leq 1. \tag{7}$$

By the fact that the geometric mean of a set of non-negative numbers is at most their arithmetic mean, we have:

$$\begin{aligned}\sum_{S_i \in \mathcal{Q}} h(S_i) &\geq |\mathcal{Q}| \cdot \Big(\prod_{S_i \in \mathcal{Q}} h(S_i)\Big)^{\frac{1}{|\mathcal{Q}|}} \\ &= |\mathcal{Q}| \cdot \big(2^{\sum_{S_i \in \mathcal{Q}} -depth(S_i)}\big)^{\frac{1}{|\mathcal{Q}|}} \\ &= \frac{n}{2}\big(2^{\sum_{S_i \in \mathcal{Q}} -depth(S_i)}\big)^{\frac{2}{n}}.\end{aligned}$$

Therefore, by Equation (7),

$$\frac{n}{2}\big(2^{\sum_{S_i \in \mathcal{Q}} -depth(S_i)}\big)^{\frac{2}{n}} \leq 1.$$

Dividing both sides by $\frac{n}{2}$, taking the logarithm and then multiplying by $\frac{n}{2}$ on both sides, we have:

$$\begin{aligned}\sum_{S_i \in \mathcal{Q}} -depth(S_i) &\leq \frac{n}{2}\log\frac{2}{n} \\ &= -\frac{n}{2}\log\frac{n}{2}.\end{aligned}$$ □

THEOREM 9. *The total number of leaves in a query tree with height k and n black leaves is at most $\frac{n}{2}(k+2-\log n)$.*

PROOF. Suppose T is the largest query tree with n black leaves. We construct the set of subtrees $\mathcal{Q}$ as in (6). Then the height of each subtree S_i in $\mathcal{Q}$ is at most $k - depth(S_i)$, since the height of the T is at most k. By Lemma 3, the number of leaves in S_i is therefore at most $k - depth(S_i) + 1$. Summing over all the subtrees in $\mathcal{Q}$, the total number of leaves, denoted by $L(\mathcal{Q})$, is given by:

$$\begin{aligned}L(\mathcal{Q}) &\leq \sum_{S_i \in \mathcal{Q}} ((k - depth(S_i)) + 1) \\ &= |\mathcal{Q}|(k+1) - \sum_{S_i \in \mathcal{Q}} depth(S_i) \\ &= \frac{n}{2}(k+1) - \sum_{S_i \in \mathcal{Q}} depth(S_i) \\ &\leq \frac{n}{2}(k+1) - \frac{n}{2}\log\frac{n}{2}, \text{ by Lemma 7,} \\ &= \frac{n}{2}(k+1-\log\frac{n}{2}).\end{aligned}$$ □