OPT: Optimal Protocol Tree for Efficient Tag Identification in Dense RFID Systems*

Girish Khandelwal Aylin Yener Min Chen
Wireless Communications and Networking Laboratory
Electrical Engineering Department
The Pennsylvania State University, University Park, PA 16802
guk109@psu.edu *yener@ee.psu.edu* *muc163@psu.edu*

Abstract— **We propose a novel collision resolution scheme termed the Optimal Protocol Tree (OPT), which is based on the tree search algorithm for RFID systems. The basic principle of OPT relies on taking advantage of the similarities in the identification strings of different tags, having the reader prompt the tags to send only the mutually exclusive sub-portion of their identification strings. The aim of OPT is to significantly reduce the total identification time, in order to render the deployment of dense RFID systems feasible. Simulation results are presented to demonstrate the performance of OPT, and the considerable improvement it provides with respect to the existing tree search protocols.**

I. Introduction

Radio Frequency Identification (RFID) is continuing to be a promising technology, with recent advances in low operating cost, ease of deployment, reliable data collection, etc. [1], [2]. The emergence of new applications mandates the future RFID systems would be highly dense and mobile. It is likely that multiple tags with unique identities move in the reader field at once, and communicate with the reader simultaneously, which will cause *tag-collisions* in RFID systems. An intense effort is now underway to design intelligent communication protocols to confront the tag-collision problems for RFID systems.

Up to date, a number of attempts have been made targeting collision-free data transmission [3]–[5]. Among tree search methods, Binary Tree Search [6] and Query Tree [7] have been suggested specifically for collision resolution in RFID systems. In Binary Tree Search protocol [6], tags continue to transmit the remainder of their Electronic Product Code (EPC) strings if their last transmitted bit matches with the one transmitted by the reader in the current bit interval, otherwise the tags get muted. Since EPC strings are unique for different tags, gradually all tags except one are muted, and at the end of an identification period, one tag is read successfully. Then another identification period starts all over again to identify the next tag. Query Tree [7] is also based on queries and responses, but the length of the reader queries changes. Specifically, in Query Tree algorithm, the reader broadcasts variable length queries, and if the query string matches with the tags' EPC prefix, they respond with their EPC and CRC strings in the next bit interval. If the reader observes a collision, it will refine the query and prompt the tags again till all tags are identified. Clearly, the length of the new query is never less than any previous query as the identification process unfolds. Both of these tree-based methods perform well only when the number of tags is relatively small, since each tag transmits its entire information bit string to the reader to build a single identity. If a large number of tags are present, the total identification time increases to the point that the delay is not acceptable for practical implementations.

The motivation of this work is to design a transmission scheme that provides significant improvement to the total identification time. Furthermore, we aim to accomplish this with minimal additional complexity. To that end, we propose the Optimal Protocol Tree (OPT), as an alternative to the tree-based protocols previously proposed for RFID systems. OPT is a novel strategy for tag identification in dense RFID systems, and takes advantage of varying degrees of inherent correlatedness in the EPC strings of the tags during the identification process. Note that the structure of the EPC code [8] is such that a minimum of 17 bits characterize 'Version Number' and 'Manufacturer Number', which is likely to be identical in tags in certain applications, such as in a factory production environment, where EPCs are attached to the products.

Operationally, OPT successfully identifies one tag in each identification period while muting the others; then it intelligently reactivates only a subset of the muted tags and invokes transmissions of only the un-transmitted portion of their EPCs in the ensuing identification periods, until successful identifying all tags. The transmissions of these mutually exclusive incremental sub-portions of the EPC in subsequent identification periods, as against the re-transmission of the entire EPC after collisions in other contemporary multiple access protocols, provides defining improvements. In this paper, we describe the protocol, present its performance analysis and demonstrate that considerable reduction of the total identification time in dense RFID systems can be obtained by OPT.

II. System Description

We consider a dense and stationary RFID system as shown in Figure 1, which consists of a reader and K *passive* tags [2] each of whom has a data string including L bits EPC

*This research is supported in part by Techcollaborative Round 11 project "Design of Efficient RFID Systems".

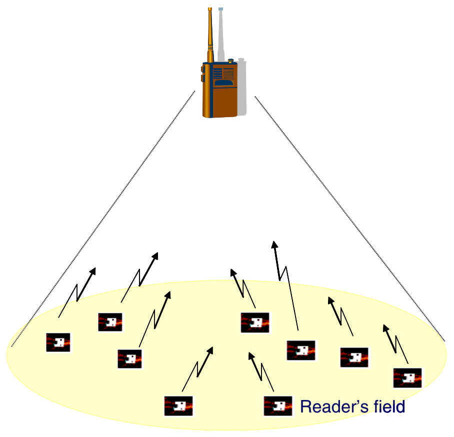


Figure 1. RFID System : Typical multiple access setup

and l_c bits CRC. Assume there is single half-duplex error-free[1] communication channel between the reader and the tags. The reader begins by broadcasting 'Reset' and 'Calibration Signals' at the beginning of an identification process, to energize and synchronize the tags [6]. We assume that once the identification process has started, energized tags remain synchronized to data bit boundaries and they remain within the reader's range. The *singulation period* is defined as the period where the reader and the tag(s) exchange data bits and one tag is identified successfully [6]. We also define the *identification process* which consists of the entire K singulation periods, i.e., the total identification time.

As proposed in [6], the reader uses three symbols to communicate with tags: 'binary 0', 'binary 1' and 'Null' for data transmission and control command. The reader and the tags communicate with each other by exchanging one bit in a data bit interval of 12.5μs in half duplex mode: the reader transmits in the former portion and the tags in the latter portion of that interval. The transmission patterns are explained in detail in [6]. We differentiate the *collision interval*, where more than one tag transmits both ('0' and '1') types of data bits in the same bit duration, from the *contention interval*, where more than one tag transmits the same ('0' or '1') bit in the same interval. Furthermore, after the transmission of the EPC string, tags also transmit the CRC string. The reader does a CRC check and the successful reception of an EPC are acknowledged, which singulates an active tag. We point out that in view of our assumptions, we have ruled out the possibility of a CRC error, i.e., a tag will always be successfully identified in every singulation period.

We assume that the reader has sufficiently large memory and computational power, whereas the passive tags have very limited capabilities. We emphasize that tags cannot exchange messages with each other and they can neither track any statistics of the channel nor do they have the mechanism for detecting collisions on the channel. It should also be noted that the reader can detect whether there is a collision, however, it does not attempt to estimate the number of tags that collide.

[1]The received SNR is shown to be high enough to justify this assumption with passive tags communicating in a short range in [9].

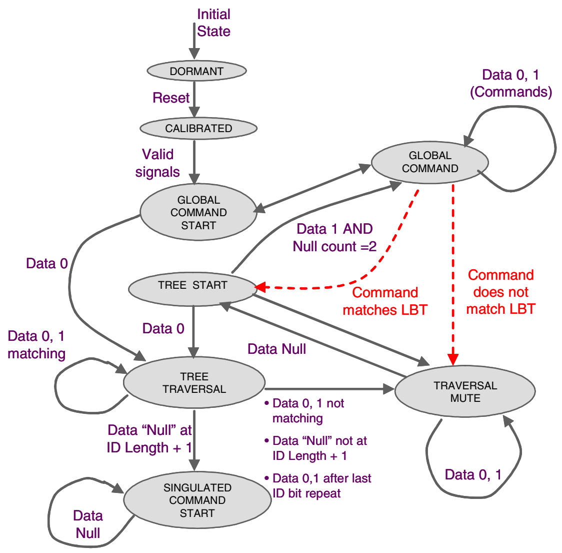


Figure 2. Modification to the tag state machine for implementing OPT

III. Optimal Protocol Tree

In this section, we propose Optimal Protocol Tree (OPT). In essence, OPT discards the inefficiencies of the Binary Tree Search and utilizes the similarities among the EPCs of the tags.

The principle of OPT is that, the reader buffers the already transmitted data bits by each tag, and uses it later to concatenate with the other portions of the EPC string of the individual tag, to build each EPC string. Thus, OPT reduces the re-transmission of the partial strings which are common to multiple tags, and hence the total identification time. We describe the mechanism of OPT in the following with the assistance of the state diagram in Figure 2. We have introduced two additional transitions (in dash '- -') to the already specified tag state machine in [6]. Section A–C below describes necessary components of OPT, while Sections D and E explain the protocol.

A. LBT (Last Bit Transmitted) Register

Each tag implements an 8-bit register for temporarily storing the index of the last bit it transmits before it gets muted. The 8-bit register is sufficient to store the bit index of future 256-bit EPC and has a minimal impact on the cost of the tags.

At the beginning of the identification process, tags must initialize the LBT to decimal '1' and increment it by '1' synchronously with the receptions of a positive acknowledgement, i.e., a matched bit from the reader. In case of negative acknowledgement, i.e., an unmatched bit, tags locks the value in their LBT registers and transition into the 'Traversal Mute' state. Tags that transition back into the 'Tree Traversal' state in subsequent singulation periods unlock the LBT register, and always increment the previously stored value in the LBT by '1' for the first data bit transmitted, and subsequent increments take place at the reception of the positive acknowledgments from the reader, as explained earlier. Tags in the 'Traversal Mute' state do not modify the contents of the LBT register. For example, if a tag transmits its 19th data bit and observes that the reader's next data bit does not match, it stores '1001 0011' in its LBT register and transitions into the 'Traversal Mute' state.

B. LIFO (Last In First Out) Queue

Simultaneous reception of '0' and '1' from different tags in the latter part of the 12.5μs bit interval results in a collision. The detection of a collision implies that one or more tags will not be positively acknowledged by the reader in the next bit interval and hence these tags will get muted. The reader will store the index of these collisions in the LIFO queue, whose content in the top-most cell corresponds to the last entry updated by the reader. For instance, if in the first singulation period, the reader observes collisions in the 1st, 4th and 7th bits of the EPC strings in sequence, values of $1, 4$ and 7 will be each stored in a cell in the LIFO queue from the bottom to the top. In the second singulation period, the value 7, which is in the top-most cell of LIFO queue is emptied since the transmission will start from the 8th bit of the EPC code, and any collisions observed in this period will be updated accordingly in the LIFO queue. Assume collisions occur in the 24th and 32th bits of EPC string in this period, then the contents in the LIFO queue will be updated as $1, 4, 24$ and 32 from the bottom to the top.

In general, the reader will observe these collision patterns during the transmission of only the EPC portions by the tags since the singulation of a tag is always accomplished before the onset of the CRC transmission within a singulation period. In view of this, the maximum value that will be stored in the LIFO queue at any instant is limited by the size of the EPC. If the LIFO queue at the end of a singulation period is empty, it implies that all the K tags have been identified, and the identification process is complete. Thus, the reader does not require the tag count in advance.

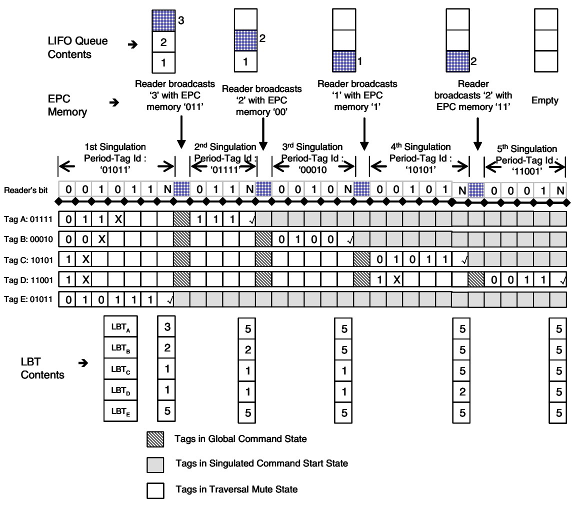
C. EPC Buffer Memory

This refers to a buffer memory that the reader uses to store the EPC string of the last successfully identified tag.

D. Working Principle of OPT

As shown in Figure 2, the reader completes a singulation period by broadcasting a data 'Null'. It serves as a positive acknowledgment to the tag in the 'Tree Traversal' about its successful identification, and transitions it into 'Singulated Start Command' state. It also transitions the other tags in the 'Traversal Mute' state into the 'Tree Start' state. At this instant, if the next data bit broadcasted by the reader is '0', all tags in the 'Tree Start' state transition into the 'Tree Traversal' state. Otherwise, another 'Null' followed by a '1' is broadcasted by the reader, and all tags in the 'Tree Start' state transition into the 'Global Command' state. These state machine transitions are already provided in [6].

Once in the 'Global Command' state, the reader can communicate with the tags by broadcasting commands formed using a set of data bits. The Binary Tree Search Protocol in [6] has defined the 8-bit commands followed by a parity bit,



LIFO Queue Contents
EPC Memory
Reader broadcasts '3' with EPC memory '011'
Reader broadcasts '2' with EPC memory '00'
Reader broadcasts '1' with EPC memory '1'
Reader broadcasts '2' with EPC memory '11'
Empty
1st Singulation Period-Tag Id : '01011'
2nd Singulation Period-Tag Id : '01111'
3rd Singulation Period-Tag Id : '00010'
4th Singulation Period-Tag Id : '10101'
5th Singulation Period-Tag Id : '11001'
Reader's bit
Tag A: 01111
Tag B: 00010
Tag C: 10101
Tag D: 11001
Tag E: 01011
LBT Contents
Tags in Global Command State
Tags in Singulated Command Start State
Tags in Traversal Mute State

Figure 3. Evolution of an Identification process in OPT

and those in the range 128 - 255 (1000 0000 - 1111 1111) are proposed as optional and reserved for manufacturers to implement unique features and functions. In OPT, we propose to use this set of commands for selectively transitioning the tags in the 'Global Command' state into the 'Tree Start' state. The reader uses the content in the top-most cell of the LIFO queue to form the instantaneous command, and empties that cell after the formation of the command. We propose to use 64 commands between '128' (1000 0000) and '191' (1011 1111), interpreted upon receptions by tags as 1 to 64, for 64-bit EPC, and 96 commands between '129' (1000 0001) and '224' (1110 0000), interpreted by tags as 1 to 96 for 96-bit EPC.

The interpreted command is then compared with the value stored in the LBT register by each tag. Tag(s) with matching commands, transition into the 'Tree Start' state, whereas the unmatched tags transition into the 'Traversal Mute' state as described in Figure 2. At this point, a new singulation period begins. However, at this time, tags in the 'Tree Traversal' state start with the transmission of the 'LBT+1' data bit of its EPC string, in response to the reader's first data bit '0' in this singulation period. Any collisions observed in this singulation period will be updated accordingly in the LIFO queue. In essence, the reader progressively identifies all EPCs one by one, by intelligently utilizing the contents in the LIFO queue and the EPC buffer memory.

We illustrate the identification process in OPT for 5 tags each with 5-bit EPC string in Figure 3. We observe that OPT takes only 27 bit intervals, including 'ACK' bits, as compared to 35 bit intervals used in Binary Tree Search Protocol.

E. Reader-Tag Communication Structure

We now have a close look at the structure of the Reader-to-Tag(s) communication (Figure 4). We observe that OPT incurs an overhead of 11 extra data bits ('Null' + '1' + 8-bit command + parity bit) for every singulation period, which does not exist in Binary Tree Search Protocol. This excludes the exchange of CRC bits and acknowledgment bits, which are present in both protocols. Fortunately, OPT com-

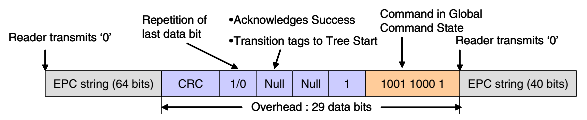


Figure 4. Reader-Tag communication structure in OPT

pensates for this overhead by having shorter length singulation periods. Specifically, we have a shorter length singulation period if tags have more than 11 common bits in the most significant portion of their EPCs. Otherwise, OPT may deteriorate the performance. In order to get around this caveat, we propose that OPT fall back to Binary Tree Search Protocol intermediately, and return to OPT once the number of common bits in the most significant portion of the EPCs increases beyond 11. Since OPT is an enhancement of original Binary Tree Search Protocol, switching between these two protocols is easily implemented with the use of proper commands at the end of each singulation period.

IV. Tag Identification Time

In this section, we discuss the performance of OPT with respect of the tag identification time. Since the duration of each bit interval in OPT is fixed as $12.5\mu s$, it suffices to find the total number of bit intervals to compute the identification time. We denote the *degree of correlation*, D, as the maximum number of bits in the most significant portion of the EPC strings, common to all K tags to be identified. Note that a subset of these tags may have a higher degree of correlation. We will restrict our analysis to RFID tags with $L = 64$ bits EPC strings, $l_c = 16$ bits CRC, and $D > 11$. When $D \leq 11$, OPT falls back to Binary Tree Search method which takes 82 bits to identify each tag. Since CRC, 'ACK' and command bits are fixed overhead, they can be counted separately.

A. Tree Structure

For easy illustration of OPT, we show a tree diagram composed of nodes and branches, where a node signifies the reader's transmitted data bit and a branch signifies tag(s) transmission. A node and a branch tied together in this order form a bit interval. When each of the tags transmits the same bit in a given bit interval, the tree (or the subtree) continues to grow deep in one direction, in sync with the EPC strings of the tags. Whenever tags' transmissions result in a collision in the latter part of a bit interval, a node diverges into two branches. Any node of the tree could have two branches at most. When branching occurs, two subtrees are formed under this node. The tree continues to grow until it hits a terminal node at the bottom of the tree, which signifies the completion of a singulation period, i.e., the successful identification of a tag. The overall depth of the tree is L, i.e., the original length of the EPC. Subsequently, the reader goes back to a node, where it last observes a collision, and begins another branching process as described previously. We revisit the example of 5 tags and extend their EPC lengths L to 64 and set $D = 59$ to illustrate the formation of the tree in Figure 5. Note that the pattern of the most significant 59 bits does

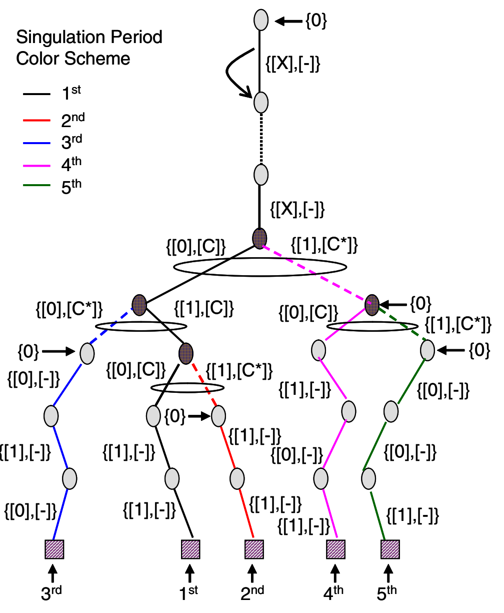


Figure 5. Tree diagram in OPT

not affect the analysis. We label each branch by $\{[\text{A}],[\text{B}]\}$, where $A \in \{0, 1\}$ denotes the bit transmitted by the tag(s). $B \in \{C, C^{*}, -\}$ provides the information about the reader's action. When a collision is observed for that node-branch pair, 'C' marks the instantaneous chosen bit/branch/direction, and 'C^{*}' corresponds to the one not chosen, whereas '$-$' means no collision is observed and no choice is made. Note that every singulation period starts with the reader broadcasting a '0', which is denoted by a horizontal arrow in the figure. The rings in the figure describe a pair of branches with collision.

From the formation of the tree, we observe that the total number of bit intervals required to identify K tags is given by the number of node-branch pairs of the tree. Terminal nodes are not included in this counting since they are the nodes without further branches. Also note that the order of the identification of tags does not change the total number of bit intervals, or the total identification time.

B. OPT in Full Binary Tree Case

We consider the case when the reader is tasked with the identification of all EPCs that can be formed with a given D. We show that a full binary tree will be formed with collision at each node beyond depth D. From Section III, we know that all tags will transmit the first D identical bits in the first singulation period. Subsequent to depth D, we will begin to observe collisions, i.e., the first branching occurs and two subtrees are formed under it. We use $D^c = L - D$ to denote the height of the subtrees. Since both subtrees will be full binary, the number of bit intervals required in each of the subtrees are the same. Thus, the number of bit intervals required for a full binary subtree with height $D^c + 1$ is one bit in addition to twice of what is required in a full binary subtree with height D^c. We note that the additional bit is spent in moving one depth lower into the tree. The branching into full subtrees of smaller height, continues to occur till the bottom of the tree is reached. Let A_{D^c} denote the number of nodes in the subtree of D^c, by employing the recursive process and using the fact that $A_1 = 1$, we have

$$A_{D^c} = 1 + 2A_{D^c-1} = 1 + 2(1 + 2A_{D^c-2}) = \cdots \quad (1)$$

$$= \sum_{i=0}^{D^c-1} 2^i = 2^{D^c} - 1 = 2^{L-D} - 1 \quad (2)$$

The total number of bit intervals for the full binary tree can be computed as

$$N_D = D + A_{D^c} = D + 2^{L-D} - 1 \quad (3)$$

Further, we calculate the average bit intervals per tag for a given $D > 11$ by adding the deterministic overhead as discussed in Section III-E. Thus, the average bit intervals per tag, E_{N_D}, for a given D can be written as

$$E_{N_D} = \frac{N_D + 18 + 29(2^{L-D} - 1)}{2^{L-D}} = 30 + \frac{D - 12}{2^{L-D}} \quad (4)$$

Here, the 18 bits are for the 2-bit 'ACK' and 16-bit CRC in the first singulation period, and the 29 bits are for the 2-bit 'ACK', 11-bit command and 16-bit CRC in the remaining singulation periods. From (4), we observe that on average OPT takes between 30 and 31 bits to identify each 64-bit EPC that can be formed for $D \in [12, 58]$.

C. Expected Total Number of Bit Intervals

We use the fact that the number of bit intervals required for identification of K tags is given by the number of node-branch pairs in the tree, to find the expected total number of bit intervals. Thus, we will find the expected number of node-branch pairs at each depth of the tree formed by K tags with D degree of correlation. The expected value for the total number of node-branch pairs can be found from the summation of the individual expectations at each depth.

First we note that for a given D, the maximum number of possible tags is 2^{L-D}. With that, the probability of any terminal node (tag) is

$$p_0 = \frac{K}{2^{(L-D)}} \quad (5)$$

With p_0 known, the probability that a node is present at the immediate next height in the tree is given by

$$p_1 = 1 - (1 - p_0)^2 \quad (6)$$

Generalizing the probability of the existence of a node at height i for $i \in [1, L - D - 1]$, we obtain

$$p_i = 1 - (1 - p_{i-1})^2 \quad (7)$$

Given that, the number of nodes at any depth is a binomial random variable with parameters $p = p_i$ ($q = 1 - p_i$) and $2^{(L-D-i)}$, and thus the expected number of the nodes at height i, X_i, is given by

$$E[X_i] = p_i \cdot 2^{L-D-i} \quad (8)$$

Consequently, the expected number of the node-branch pairs in the whole tree is

$$\sum_{i=1}^{L-D-1} E[X_i] = \sum_{i=1}^{L-D-1} p_i \cdot 2^{L-D-i} \quad (9)$$

Finally, the expected number of total bit intervals N given K and D, including the fixed D bits in the first singulation period, is

$$E[N|K, D] = D + \sum_{i=1}^{L-D-1} p_i(K, D) \cdot 2^{L-D-i} \quad (10)$$

Observe that in (10) the dependency of p_i on K and D is explicitly stated. We can compute the expected number of bits in the final implementation of OPT by adding the deterministic overhead, as explained in IV-B. The calculation of the expected total identification time is straight forward.

Lastly, we note that for mobile/dynamic RFID systems where the joint probability distribution of K and D is available, we can easily compute the total expected number of bits $E[N]$.

V. Numerical Results

We present numerical results related to the performance of OPT in this section. First, we compare the performance of OPT with Binary Tree Search Protocol [6], with the average number of bit intervals per tag as the performance metric. In each iteration of our simulation, we randomly generate K EPC strings, which are identical in their D most significant bits. In the identification process, the EPC strings are identified sequentially in an increasing order of their numerical decimal value. For a fair comparison between two protocols, we take into account the bit intervals used in state transitions and commands during the implementation of OPT. As discussed in III-E, the identification procedure falls back to Binary Tree Search, whenever contents of the top-most cell of the LIFO queue falls below 11. In such a case, the intermediate singulation period becomes a full length of 82-bit interval.

In Figure 6, we plot the average number of bit intervals per tag for varying degrees of correlation D, when K is between 1 and 100. We observe that OPT shows significant improvement over the Binary Tree Search, especially for large D. This is expected because OPT saves the re-transmission of the identical prefix bits of different tags. We even observe an improvement when the degree of correlation is 10. This is because, when unique EPCs are randomly generated for large K and $D \leq 11$, it is possible that they have a subset of tags with more than D identical bits, and any such subset will be identified utilizing OPT in successive singulation periods of shorter lengths. We note that it is also possible to have many such subsets in an identification process, and hence the identification protocol may switch between OPT and Binary Tree Search multiple times. The average bit intervals for higher values of K are plotted in Figure 7, and the trends observed for small K are visible for large K as well. We also point out that the simulation results compare closely with the values computed from (10).

Recall from (4) that OPT requires an average of 30-31 bits per tag i.e., a 62% improvement over Binary Tree Search Protocol, when it identifies a complete set for a given D. These trends are observed for $D = 57$ and $K = 128$ in Figure 6, and for $D = 54$ and $K = 1024$ in Figure 7.

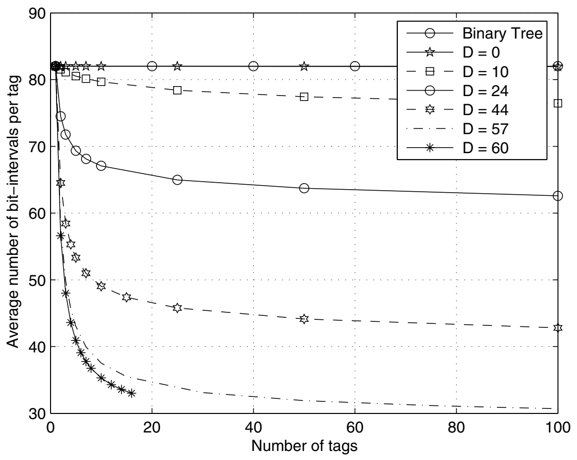


Figure 6. Trend for average number of bits as D varies, for small K.

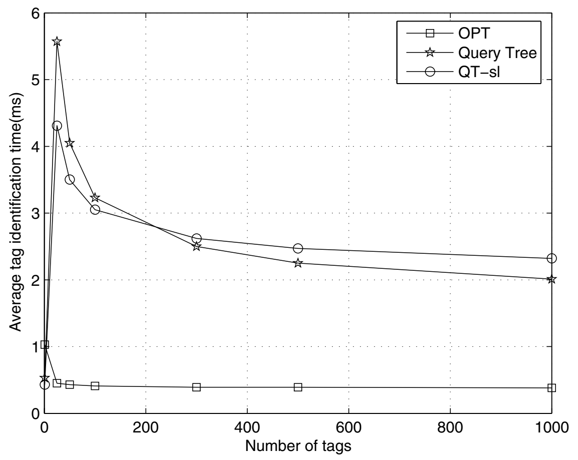


Figure 8. OPT vs. Query Tree : Average tag identification time for $D = 54$.

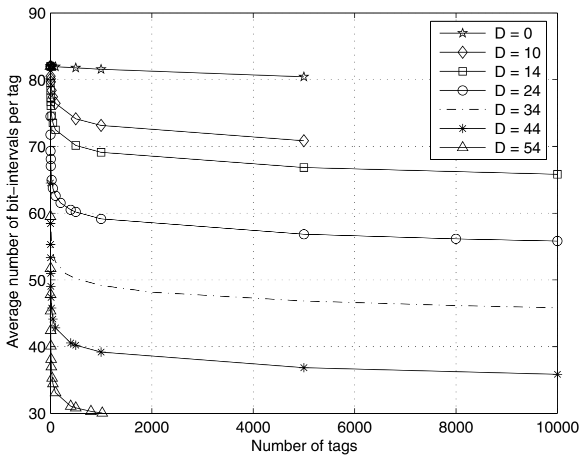


Figure 7. Trend for average number of bits as D varies, for large K.

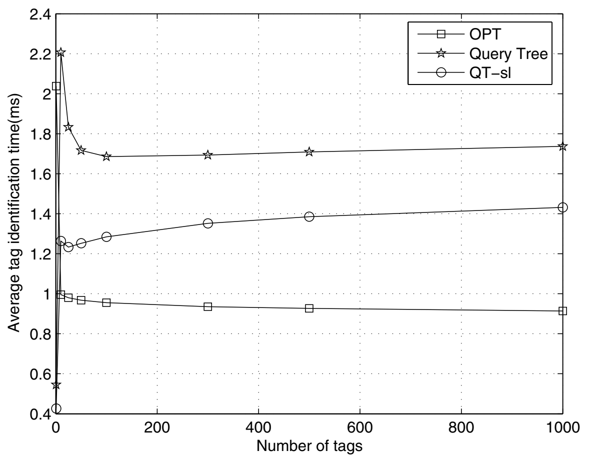


Figure 9. OPT vs. Query Tree : Average tag identification time for $D = 10$.

We also compare the performance of OPT with Query Tree, and its variant QT-sl protocols proposed in [7]. In view of the differences among these protocols, and for a fair comparison, the communication structures for Query Tree and QT-sl protocols are devised in [9]. The average tag identification time of different protocols are shown in Figure 8 and Figure 9 for $D = 54$ and $D = 10$, respectively. We observe that OPT performs significantly better than both Query Tree based protocols, even for small degree of correlation, e.g. $D = 10$, despite the fact that the performance of Query Tree improves as D decreases while the performance of OPT suffers with the decrease in D.

VI. Conclusions

In this paper, we have proposed OPT, an efficient and tree search based multiple access technique for RFID systems. We have observed that OPT outperforms Binary Tree Search and Query Tree Protocols significantly. We demonstrated analytically and numerically that OPT can significantly reduce the re-transmission between tags and readers, and hence the total identification time, a feature that is highly desirable in dense RFID systems envisioned for the near future.

References

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

[2] K. Finkenzeller. *RFID Handbook: Radio-Frequency Identification Fundamentals and Applications*. John Wiley & Sons, 2000.

[3] J. Capetanakis. Generalized TDMA: The multi-accessing tree protocol. *IEEE Transactions on Communications*, COM-27:1476–1484, October 1979.

[4] B. Tsybakov. Survey of USSR contributions to random access communications. *IEEE Transactions On Information Theory*, IT-X, No. 2:143–165, March 1985.

[5] J. Massey. Collision-resolution algorithms and random-access communications. Multi-User Communications Systems, Pages:73-99, 1981.

[6] Draft protocol specification for a 900 MHz Class 0 Radio Frequency Identification Tag. Technical report, Auto-ID Center, Feb 2003.

[7] C. Law, K. Lee, and K. Siu. Efficient memoryless protocol for tag identification. In *4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, ACM*, pages 75–84, August 2000.

[8] 13.56 MHz ISM Band Class 1 Radio Frequency Identification Tag Interface Specification: Recommended Standard, Version 1.0.0. Technical report, Auto-ID Center, May 2003.

[9] G. Khandelwal. Efficient design of dense and time constrained RFID systems. M.S. Thesis, The Pennsylvania State University, August 2005.