Identifying State-free Networked Tags

Min Chen Shigang Chen

Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL 32611, USA
Email:{min, sgchen}@cise.ufl.edu

***Abstract*—Traditional radio frequency identification (RFID) technologies allow tags to communicate with a reader but not among themselves. By enabling peer communications between nearby tags, the emerging networked tags represent a fundamental enhancement to today's RFID systems. They support applications in previously infeasible scenarios where the readers cannot cover all tags due to cost or physical limitations. This paper is the first study on identifying state-free networked tags, which is a basic, fundamental function for most tagged systems. To prolong the lifetime of networked tags and make identification protocols scalable to large systems, energy efficiency and time efficiency are most critical. Our investigation reveals that the traditional contention-based protocol design will incur too much energy overhead in multihop tag systems. Surprisingly, a reader-coordinated design that significantly serializes tag transmissions performs much better. In addition, we show that load balancing is important in reducing the worst-case energy cost to the tags, and we present a solution based on serial numbers.**

I. Introduction

RFID (radio frequency identification) tags are becoming ubiquitously available as they make their way into retail products, library books, debit cards, passports, driver licenses, car plates, medical devices, etc [1]–[3]. Each tag carries a unique ID for identifying the object it is attached to [4], [5]. Today's commercial tags can be classified into three categories: (1) *passive tags*, which are powered by the radio wave from an RFID reader and communicate with the reader through backscattering; (2) *active tags*, which are powered by their own energy sources; and (3) *semi-active tags*, which use internal energy sources to power their circuits while communicating with the reader through backscattering. All these tags can only communicate with RFID readers but not amongst themselves. Therefore the coverage of an RFID system is strictly limited to the communication range of its readers.

The emerging *networked tags* promise to bring a fundamental enhancement by enabling tags to communicate with each other. An example of such tags is a new class of ultra-low-power energy-harvesting networked tags designed and prototyped at Columbia University [6], [7]. Tagged objects that are not traditionally networked, e.g., warehouse products, books, furniture, and clothing, can now form a network [8], which provides great flexibility in applications. Consider a large warehouse where a great number of readers and antennas must be deployed to provide full coverage. Not only is this costly, but obstacles and piles of tagged objects may prevent signals from penetrating into every corner of the deployment, causing a reader to fail in accessing some of the tags. This problem will be solved if the tags can relay transmissions toward the otherwise-inaccessible reader. Similar situations arise in retail stores (or libraries, chicken factories, livestock ranches) where continuously packed products (or books, animals) provide an ideal environment for tag-to-tag communications, which can drastically alleviate the coverage requirement for reader deployment, reduce the number of readers needed, and more importantly provide additional assurance for tag access under practical conditions. For example, a new shelf can be immediately installed as needed in a location without prior reader coverage as long as any part of the shelf is within tag-to-tag communication range of another shelf. More broadly, with emergence of the Internet of Things [8], we envision that networked tags will play an important role as an enhancement to the current RFID technologies.

The new feature of networked tags opens up new research opportunities. This paper focuses on the tag identification problem, which is to collect the IDs of all tags in a system. This is the most fundamental problem for RFID systems, but has not been studied in the context of networked tags, where one can take advantage of the networking capability to facilitate the ID collection process, e.g., collecting IDs of tags that are not within the reader's coverage. Beyond the coverage of the readers, the networked tags are powered by batteries or rechargeable energy sources that opportunistically harvest solar, piezoelectric, or thermal energy from surrounding environment [7], [9]. Energy efficiency is a first-order performance criterion for operations carried out by networked tags.

There are two types of networked tags. The *stateful* networked tags maintain network state such as neighbors and routing tables and update the information to keep it up-to-date. These tags resemble the nodes in a typical sensor network. On the contrary, for the purpose of energy conservation, the *state-free* tags do not maintain any network state prior to operation, which makes them different from traditional networks, including sensor networks — virtually all literature on data-collecting sensor networks assume the stateful model, where the sensor nodes maintain information about who are their neighbors and/or how to route data in the network. This paper considers state-free networked tags, not only because there is little prior work on this type of networked nodes, but also because it makes more sense for the tag identification problem: First, establishing neighborship and then routing tables across the network is expensive and may incur much more overhead than tag identification itself, which only requires each tag to

deliver one number (its ID) to the reader. Second, maintaining the neighbor relationship and updating the routing tables (as tags may move between operations) require frequent network-wide communications, which is not worthwhile for infrequent operation of tag identification.

It is challenging to design an identification protocol for state-free networked tags. First, because power is a scarce resource for tags, the protocol must be energy-efficient in order to reduce the risk of network failure caused by energy depletion. Second, we should also make the protocol time-efficient so that it can scale to a large tag system where the communication channel works at a very low rate for energy conservation. Third, in order to eliminate overhead of state maintenance and thus conserve energy, tags are assumed to be state-free, which means that they do not know who are their neighbors and there is no existing routing structure for them to send IDs to the reader.

To the best of our knowledge, this is the first work that tries to solve the problem of collecting IDs of state-free networked tags. The existing RFID identification [10]–[13] protocols cannot be applied because they assume that the readers can reach all tags directly. We have to make a fundamental shift in the protocol design for networked tags. This paper presents two solutions: a contention-based ID collection protocol and a serialized ID collection protocol. Our investigation reveals an interesting result that the traditional contention-based protocol design will incur too much overhead in multihop networked tag systems due to progressively increased collision in the network towards a reader, which results in excessive energy cost. Surprisingly, a reader-coordinated design that attempts to serialize tag transmissions performs much better. Moreover, we discover that load balancing is critical in controlling the worst-case energy cost incurring to the tags. We find that the worst-case energy cost is high for both contention-based and serialized protocol designs due to imbalanced load in the network. For the serialized ID collection protocol, however, we are able to provide a solution based on serial numbers that balance the load and reduce the worst-case energy cost. With serialization and load balancing, we show through simulations that the transmission and receiving overheads per tag are reduced by up to 90.0% and 88.8% respectively, with a comparable protocol execution time.

II. System Model and Problem Statement

A. Networked Tag System

We consider a reader and a large number of objects, each of which is attached with a tag. We will use tag, node and networked tag interchangeably in the sequel. Each tag has a unique ID that identifies the object it is attached to. The reader also has a unique ID that differentiates itself from the tags.

A networked tag system is different from a traditional RFID system with a fundamental change: Tags near each other can directly communicate. This capability allows a multihop network to be formed amongst the tags. Developed at Columbia University recently [7], prototype networked tags can communicate using variants of CSMA and slotted ALOHA. The transmission range of inter-tag communications is usually short, about 1 to 10 meters [8]. But the reader is a more powerful device, and its transmission range can be much larger. Tags that can perform direct two-way communicate with a node form the *neighborhood* of the node.

Networked tags are expected to carry sufficient internal energy for long-term operations or have the capability of harvesting energy from the environment where they are deployed. Tags of the highest energy demand are located in the reader's neighborhood (i.e., coverage area) because they have to relay the information from all other tags as the data converge towards the reader. Fortunately, these tags can be powered by the reader's radio waves, similar to what today's passive RFID tags do; their energy supply is ensured. In contrast, tags that are beyond the reader's coverage need to use their own energy. The operations of these tags must be made energy-efficient.

The reader and the tags in the system form a connected network. In other words, there exists at least one path between the reader and any tag such that they can communicate by transmitting data along that path. Tags that are not reachable from the reader are not considered to be in the system.

B. Problem Statement

The problem of tag identification is for a reader to collect IDs from all networked tags that can be reached by the reader over multiple hops with the help of intermediate tags relaying the IDs of tags that are not in the immediate coverage area of the reader. Our goal is to develop tag identification protocols that are efficient in terms of energy cost and protocol execution time. We will consider both average energy cost per tag and maximum energy cost among all tags in the system. The average energy cost is an overall measurement of energy drain across the whole system, and the maximum energy cost is a measurement for the worst hot spot which may cause power-exhausted tags and network partition.

C. System Model

What makes tags attractive is their simplicity. There is no specification on how simple future networked tag tags should be, but it is safe to say that we will always prefer protocol designs that achieve comparable efficiency with less hardware requirement. Generally speaking, each tag has very limited energy, memory, and computing resources. In this paper, we do not require tags to implement GPS, any localization mechanism, or other complex functions. We consider state-free tags, which do not spend energy in maintaining any state information prior to operation.

Since each tag is only equipped with a single transceiver, it cannot perform transmission and reception simultaneously. Assume that the reader and tags cannot resolve collided signals. Therefore, a node can successfully receive the transmission only if there is only one neighbor transmitting.

For state-free tags, there is no mechanism (such as frequent beacon exchange between neighbors) that keeps track of the changes in network topology in real time. We assume that the tags are stationary during the operation of tag identification.

For example, in a warehouse, the daily tag identification may be performed automatically in after-work hours when objects are not moved around. During the daytime between the previous identification and the next one, objects can still be freely moved around. In case that the identification operation needs to be performed during the daytime, we need to design a protocol that takes as little time as possible to avoid significant interruption to other warehouse operations due to the stationary requirement at the time of identification.

To conserve energy, networked tags are likely configured to sleep and wait up periodically for operations. After wake-up, a tag will listen for a request broadcast from the reader into the network, which either puts the tag back to sleep or asks the tag to participate in an operation such as reporting its ID. The broadcast request will serve the purpose of loosely re-synchronizing the tag clock. The reader will time its next request a little later than the timeout period set by the tags to compensate for the clock drift and the clock difference at the tags due to broadcast delay. The exact sleep time of the tags and the inter-request interval of the reader should be set empirically based on application needs and physical parameters of the tags.

III. Contention-based ID Collection Protocol (CICP) for networked tag Systems

We are not aware of any existing data collection protocol specifically designed for the state-free model which makes sense in the domain of tags but was not adopted in the mainstream literature of sensor networks or other types of wireless systems. However, it is not hard to design an ID collection protocol for networked tags based on techniques known in existing wireless systems. For example, in this section, we will follow an obvious design path based on broadcast, spanning tree and contention-based transmission. The resulting protocol will be used as a benchmark for performance comparison (since there is no prior work on identifying networked tags). In the next section we will point out that the obvious techniques are however inefficient and other less-obvious design choices can produce much better performance.

A. Motivation

One straightforward approach for tags to deliver their IDs to the reader is through flooding: As each tag broadcasts its ID and every other ID it receives for the first time into its neighborhood, the IDs will eventually reach the reader. However, flooding causes a lot of communication overhead. In addition, each tag has to store the IDs that it has received in order to avoid duplicate broadcast. Due to the nature of flooding, it means that eventually each tag will store all IDs in the system, which demands too much memory.

Another approach is to ask tags to discover their neighbors and run a routing protocol to form routing paths towards the reader right before sending the IDs (even though the tags are state-free prior to operation). However, as the number of neighbors can be in hundreds in a packed system, the overhead of doing so will be high, considering that only one ID per tag will be delivered.

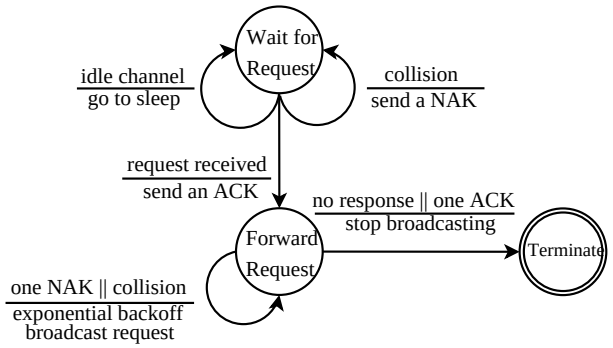

Fig. 1: State transition diagram of the RBP protocol. Each circle is a state, and each arrow is a transition, where the event triggering the transition is above the line and the action is below the line.

As the above two approaches do not work well, our idea is to establish routing paths for free. For a reader to begin the tag identification process, it needs to broadcast a request to all tags. We can make extra use of this network-wide broadcast to piggyback the function of establishing a spanning tree that covers all tags, with the reader at the root of the tree. This tree will be used for transmitting the IDs to the reader. We use the ALOHA protocol to resolve the contention among concurrent transmissions made by close-by tags.

B. Request Broadcast Protocol (RBP)

The classical broadcast protocol is for each node to transmit a message when it receives the message for the first time. But it becomes more complicated to guarantee that all nodes receive the message: If each node knows its neighbors, it may keep transmitting the message until receiving acknowledgements from all neighbors. However, more care must be taken if the nodes do not know their neighbors. Below we briefly describe a request broadcast protocol (RBP) that guarantees delivering a request from the reader to all state-free tags.

To initiate tag identification, the reader broadcasts a request notifying the tags to report their IDs. The request initially carries the reader's ID, which will later be replaced with a tag's ID when the tag forwards the request to others. The state transition diagram of the protocol is depicted in Fig. 1, which is explained below.

State of Waiting for Request: Each tag begins in this state and takes action based on one of three possible events.
(1) *Idle Channel*: The channel is idle, i.e., no neighbor is transmitting anything.
(2) *Request Received*: Only one neighbor is forwarding the request, so the tag can receive the request correctly.
(3) *Collision*: Multiple neighbors are forwarding the request, resulting in a collision.

In event (1), the tag does nothing. In event (2), the tag will reply with an acknowledgement (ACK) to inform the sender that it has successfully received the request. Meanwhile, it

extracts the ID from the request and saves it as its parent. After that, it moves to the state of Forwarding Request. As we will see shortly, it is not important whether the ACK is correctly received by the sender of the request or not. The sender will know that all its neighbors have received the request when it does not hear any response (since the neighbors all move to the state of Forwarding Request). In event (3), the tag cannot resolve the collided. It sends a negative acknowledge (NAK) and stays in the state of Waiting for Request.

State of Forwarding Request: To ensure that the request will be propagated across the network, each tag having received the request will keep broadcasting it with exponential backoff upon collision until all its neighbors receive the request. Each time after the tag broadcasts the request (which carries the tag's ID), there are three possible events:
(1) *No Response*: No response is received from any neighbor.
(2) *One ACK/NAK*: Only one ACK/NAK response is received.
(3) *Collision*: Multiple ACK/NAK responses are sent by the neighbors, leading to a collision.

Recall that any neighbor in the state of Wait for Request will respond either ACK or NAK regardless of whether it can successfully receive the request or not. Event (1) must mean that all the neighbors have already received the request and moved to other states. In this case, the tag does not need to broadcast the request any more. If no response is heard after broadcasting the request for the first time, the tag knows it has no child and it is therefore a leaf node in the spanning tree. In event (2), if a single ACK is received, the tag knows that all its neighbors now have received the request. Hence, it can stop broadcasting the request. If a single NAK is received, the tag knows that there must have been collision at a neighbor, which did not receive the request successfully. Therefore, the tag should perform an exponential backoff to avoid continuous collision in the channel. In event (3), the tag cannot resolve the received ACK/NAK correctly and it also performs an exponential backoff. As an example, Fig. 2 illustrates the spanning tree built in a networked tag system after it executes RBP, where the reader has an ID 0.

The wireless transmissions in RBP can be implemented either based on unslotted ALOHA or based on slotted ALOHA. Slotted ALOHA is more efficient but requires the tags to synchronize their slots. When the reader transmits its request to nodes in its neighborhood, the preamble of the transmission provides the clock and slot synchronization. Similarly, when a distant tag receives the request for the first time from another tag, the preamble of the latter synchronizes the clock and slot.

Theorem 1: Every tag will receive a copy of the request sent out by the reader under RBP.

Proof: To prove by contradiction, let's assume there exists a tag T that does not receive the request after executing RBP. It must be true that none of its neighbors has received the request. Otherwise, according to the protocol, any neighbor having received the request would continue broadcasting the request until T receives it and acknowledges its receipt — each time the request is transmitted, if T does not receive the request successfully, it will respond NACK, causing the sender to retransmit. By the same token, the neighbors of any T's neighbor must not receive the request. Applying this argument recursively, all nodes reachable from T must not receive the request. By the assumption that the network is connected, at least one neighbor T' of the reader is reachable from T. Therefore, T' must not receive the request. This contradicts to the fact that T' is located in the reader's coverage area and should receive the request at the very beginning when the reader broadcasts the request for the first time. Therefore, the theorem must hold. ■

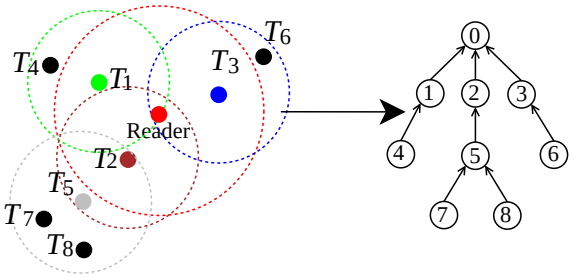

Fig. 2: An example of the spanning tree built by RBP, where the ID of tag T_i is assumed to be i. Each dotted circle on the left gives the neighbors of a tag at the center of the circle.

C. ID Collection Protocol (ICP)

When a tag transmits its ID, it will include its parent's ID in the message, such that the parent node will receive it while other neighbors will discard the message. This unicast transmission is performed based on the classical ALOHA with acknowledgement and exponential backoff to resolve collision. The parent node will forward the received ID to its parent, and so on, until the ID reaches the reader.

The execution of ICP is performed in parallel with RBP: Once a tag knows its parent ID from RBP, it will begin transmitting its ID to the parent. When a tag needs to forward both an ID for ICP and a request for RBP, we give priority to ID forwarding because it is easier for unicast to complete.

Theorem 2: The reader will receive the IDs of all tags in the system after the execution of ICP.

Proof: From Theorem 1, each tag is guaranteed to receive the request and therefore find a parent (from which the request is received). Consider an arbitrary tag T. According to the design of ICP, the ID of T will be sent to its parent until positively acknowledged. The parent will forward the ID to its parent, and as this process repeats, the ID will eventually reach the reader at the root of the spanning tree. ■

IV. Serialized ID Collection Protocol (SICP)

A. Motivation

The contention-based protocol ICP allows parallel transmissions by non-interfering tags through spatial channel reuse. In the conventional wisdom, this is an advantage. However, we find in our simulations that the contention-based protocol performs poorly for tag identification. The reason is that although parallel transmissions are enabled among the tags

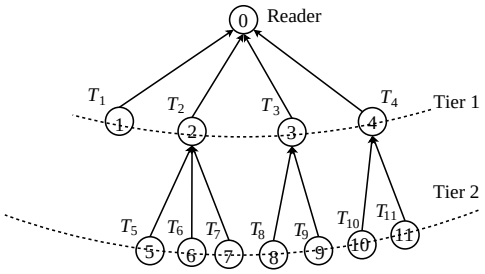

Fig. 3: At any time, only one node is active in collecting IDs from its lower-tier neighbors.

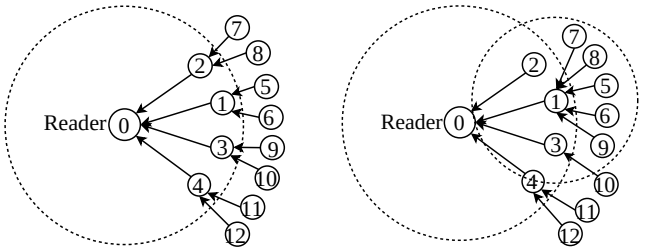

Fig. 4: *Left Plot*: a roughly balanced spanning tree; *Right Plot*: a biased spanning tree, where tag 1 delivers its ID to the reader first and a large number of nodes in its neighborhood chooses it as their parent, causing a biased tree. Each arrow represents a child-parent relationship.

in the network, the reader can only take one ID at a time. Essentially, the operation of ID collection is serialized at the reader, regardless of how much parallelism is achieved inside the network of tags. Furthermore, the parallelism is actually harmful because the more the IDs are crowded to the reader in parallel, the more the contention is caused at the reader, resulting in many failed transmissions due to collision, which translates into high energy cost and long protocol execution time. When tags are densely deployed, this problem can severely degrade the system performance. With this observation, we take a different design path by trying to partially serialize the tag transmissions, such that only a (small) portion of tags will attempt to transmit at any time. By lessening the level of contention, we see a drastic performance improvement. Another serious problem of RBP/ICP is that the spanning tree is unbalanced, causing significantly higher energy expenditure by some tags than others. This problem of biased energy consumption and a solution will be explained in details later.

B. Overview

We give an overview of our serialized protocol, SICP. The reader begins by collecting IDs in its neighborhood using framed ALOHA. An illustrative example is shown in Figure 3, where the reader collects the IDs from neighbors T_1 through T_4 (which form tier 1), while all other nodes stay idle. When the reader receives a tag's ID (say, T_2) free of collision, T_2 must be the only tag that is transmitting in the whole network. It also means that other neighbors of T_2 can hear the transmission free of collision. These tier-2 nodes, T_5 through T_7, set T_2 as their parent.

After collecting all tier-1 IDs, the reader sequentially informs each tier-1 node to further collect IDs from its children. For example, when the reader informs T_2 to do so, all other tier-1 nodes will stay silent. As T_2 sends out a request for IDs, only its children (T_5, T_6 and T_7) will respond. The same process as described in the previous paragraph will repeat; only this time T_2 takes the role of the reader.

After T_2 collects the IDs of all its children, it will forward the IDs to the reader, which will then move to the next tier-1 node. Once it exhausts all tier-1 nodes, it will move to tier-2 nodes, one by one and tier by tier, until the IDs of all nodes in the network are collected.

Below we will first introduce the problem of biased energy consumption, give a solution, and then describe recursive serialization.

C. Biased Energy Consumption

When a tag is transmitting its ID to the reader, its neighbors outside of the reader's coverage can overhear the ID. They may use this tag as their parent. As illustrated in the left plot of Fig. 4, we prefer a roughly balanced spanning tree where each node serves as the parent for a similar number of children. In reality, however, a tag that delivers its ID to the reader early on will tend to have many more children. An example is given in the right plot of Fig. 4. Suppose tag 1 transmits its ID to the reader first. Overhearing its ID, tags 5-9 will pick tag 1 as their parent. When tag 2 transmits its ID at a later time, no tag will be left to choose tag 2 as parent even though tags 7-8 are in the range of tag 2 — recall that they have already chosen tag 1. In this case, tag 1 will have to forward more IDs, resulting in quicker energy drain than others. The severity of the problem grows rapidly with an increasing number of tiers because the numerous children of tag 1 tend to acquire even more numerous children of their own and those IDs will pass through tag 1 to the reader.

Uneven energy consumption causes some tags to run out of energy earlier, which can result in network partition. The same problem also exists for RBP/ICP where tags that receive and forward the request early on during the network-wide broadcast may end up with a large number of children.

To alleviate this problem, we observe that a tag may overhear multiple ID transmissions over time and thus have multiple candidates to choose its parent from, as shown by Fig. 5 where T may choose its parent from three tier-1 nodes. Ideally, the tag should choose its parent uniformly at random from the candidates. However, because of collision, each candidate may have to retransmit its ID for a different number of times before the reader successfully receives it. To avoid giving more chance to a candidate that retransmits its ID more times, the tag may keep the IDs of all known candidates to filter out duplicate overhearing. However, a serious drawback of this approach is that the memory cost can be high if a tag has numerous candidates for its parent in a system where

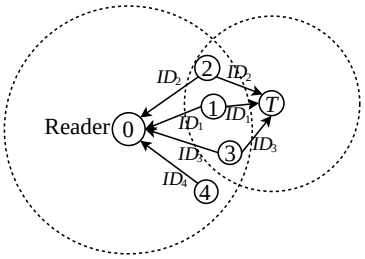

Fig. 5: A tag may choose its parent from multiple candidates. Arrows represent ID transmissions (or broadcast).

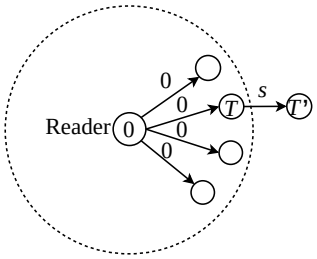

Fig. 6: T' sets T as its parent.

tagged objects are packed tightly together. We want to point out that typical tags have very limited memory.

D. Serial Numbers

We propose a solution to biased energy consumption based on serial numbers. In our protocol, each tag will be dynamically assigned a serial number from 1 to N, where N is the number of tags. The reader's serial number is 0.

Let's first consider the reader's neighborhood only; other tiers will be explained later. The reader initiates the protocol by broadcasting an ID collection request, carrying its serial number and a frame size f. The request is followed by a time frame of f slots. Each tag that receives the request will set the serial number 0 (i.e., the reader) as its *parent* and then randomly chooses a slot in the time frame. It waits until the chosen slot to report its ID to the reader. If only one tag selects a certain slot, its ID will be correctly received by the reader, which replies an ACK to the tag in the same slot. The ACK carries the number of IDs that the reader has successfully received so far. This number is assigned as the serial number of the tag; the number is system-wide unique due to its monotonically-increasing nature. A tag can be identified either by its ID or its assigned serial number. After receiving the ACK, we require the tag to broadcast the assigned serial number in its neighborhood. Hence, each time slot contains an ID transmission, an ACK transmission, and a serial-number transmission. If the ID transmission is collision-free, so do the other two transmissions. Even though a tag may need to retransmit its ID multiple times due to collision, it will transmit its assigned serial number once, only at the time when an ACK is received.

If the reader observes any collision in the time frame, it will broadcast another request with another time frame to collect more IDs. If no collision is observed, the reader has collected all IDs from its neighborhood and it will perform *recursive serialization* (to be discussed) to collect IDs outside of its neighborhood.

E. Parent Selection

Consider an arbitrary neighbor of T, denoted as T', which has not set its parent yet. As illustrated in Fig. 6, T' must not be in the reader's neighborhood because the tags in that neighborhood set the serial number 0 as their parent when they receive the request from the reader for the first time. When T' receives a serial number for the first time, it will set the number as its parent, which is subject to change when T' receives more serial numbers from other tags (candidates for parent). Recall that each tag broadcasts its serial number only once. This property allows us to design the following *parent selection algorithm* (PSA) which guarantees every candidate has an equal chance to be selected as the parent: Each tag maintains two values, its parent and a counter c for the number of candidates having been discovered so far. The counter is initialized to zero. Each time when T' receives a serial number s' from a neighbor, it increases c by one and then replaces the current parent with s' by a probability $\frac{1}{c}$. Using this PSA, we have the following theorem:

Theorem 3: Suppose a tag has m candidates for parent. Each candidate has an equal probability of $\frac{1}{m}$ to be chosen as the tag's parent in the end.

Proof: For the jth ($1 \leq j \leq m$) discovered candidate, it becomes the final parent only if it replaces the previously selected parent, and is never substituted by the subsequently discovered candidates. Therefore, the probability that it is chosen by the tag as the parent in the end is

$$\frac{1}{j}\prod_{l=j+1}^{m}(1-\frac{1}{l}) = \frac{1}{m}, \tag{1}$$

implying every candidate is equally likely to be the parent. □

Another advantage of using the serial number instead of ID for parent identification is that an ID — typically 96 or more bits for RFID tags — is much longer than the size of the serial number, $\lceil \log_2 N \rceil$, where N is the maximum number of tags in a system. For example, even if $N = 1,000,000$, the serial number is just 20 bits long.

F. Serialization at Tier 2

After the reader collects all IDs from its neighborhood, each tag in the neighborhood will obtain a unique serial number. Recall that these tags constitute the first tier of the network. The reader then serializes the subsequent ID collection process by sending the serial numbers of tier-1 tags one by one, in order to command the corresponding tag to collect IDs from its neighbors, with other tier-1 tags staying idle.

The reader begins by transmitting the serial number 1, together with the number s of IDs it has received so far. In response, the tag with the serial number 1, denoted as T_1, transmits an ID collection request, carrying its own serial

number 1 and a frame size f. The request causes the neighbors that are not tier-1 to finalize their parent selection; these nodes are tier-2. Note that some of them may have selected nodes other than T_1 as their parents. Hence, when a tier-2 node receives the request from T_1, only if its chosen parent matches the serial number in the request, it will transmit its ID in the subsequent time frame; otherwise, it can sleep for a duration of f slots. If T_1 correctly receives an ID in a slot from a child T_1', it increases the value of s by one and sends back an ACK with s as the serial number assigned to T_1', which in turn broadcasts its serial number and tier number (i.e., 2) in its neighborhood such that the neighbors at the next tier can discover it as one of their candidates for parent. When a tag sets (or later replaces) its parent, it also sets its tier number as the tier number of its parent plus one; it should never replace its current parent with one whose tier number is larger.

It may take T_1 multiple requests to finish reading all IDs from its children. It then forwards the IDs to the reader. After acknowledging T_1, the reader sends a command to trigger the ID collection process at the next tier-1 tag.

After the reader finishes this process with all tier-1 tags, it has collected the IDs of all tier-2 tags. The reader also has the information to construct a spanning tree covering tier-1 and tier-2 nodes, as illustrated in Fig. 3 where the assigned serial numbers are shown inside the circles.

G. Recursive Serialization

After the reader commands all tier-1 tags one by one to collect the IDs of tier-2 tags, it repeats this serialization process recursively to collect other IDs tier by tier. Suppose the reader has collected the IDs from all tags at tier 1 through tier i and the range of serial numbers at tier i is from x to y. The reader will send a command to each tier-i tag in sequence. The command includes a concatenation of the serial numbers along the path in the spanning tree from the root (excluded) to that tag, in addition to the number s of IDs that the reader has received so far. For example, for tag 7 in Fig. 3, the command will carry two serial numbers, 2 and 7. (Note that since each serial number is of fixed size, there is no ambiguity on interpreting the sequence of serial numbers.)

When the reader broadcasts the command in its neighborhood, any tag receiving the command will extract and compare the first serial number with its own. If the two serial numbers do not match, it discards the command. Otherwise, it further checks whether there are more serial numbers in the command. If so, it broadcasts the remaining command. This process repeats until a tag matches the last serial number in the command. That tag will performs ID collection in a similar way as described in Section IV-F. The collected IDs will be sent through the parent chain to the reader.

Theorem 4: The reader will receive the IDs of all tags in the system after the execution of SICP.

Proof: Proving by contradiction, we assume at least one tag T fails in delivering its ID to the reader. T must not have a parent; we again prove this by contradiction: Assume that T has a parent T'. According to the protocol, for T' to be chosen as a parent, it must either the reader or a node that has already successfully delivered its ID and subsequently broadcast its assigned serial number. Hence, it will receive a command from the reader to collect IDs from its children. After the reader sends a command to T', T' will broadcast requests, free of collision due to serialization, to children until all IDs are collected — which happens when no collision is detected in the time frame after a request. When T' receives the ID of T, if it is not the reader, it will forward the ID to the reader along the path with which its own ID has been successfully delivered, free of collision due to serialization. This contradicts to the assumption that T fails in delivering its ID to the reader. Hence, T does not have a parent.

If T does not have a parent, all of its neighbors must fail in delivering their IDs to the reader because otherwise any successful neighbor would broadcast its serial number according to the protocol, which would result in T having a parent after T receives the serial number.

If all neighbors of T fail in delivering their IDs to the reader, by the same reasoning as above, all their neighbors must fail too. Recursively applying this argument, all tags in the network must fail in delivering their IDs to the reader because the network is connected, which contradicts at least to the fact that the reader's immediate neighbors are able to send their IDs to the reader through the slotted ALOHA protocol that SICP employs. Hence, the theorem is proved. ■

H. Frame Size

When the reader or a tag tries to collect the IDs in its neighborhood, its request carries a frame size f. Let n be the number of tags that are children of the reader or tag sending the request. It is well known that the optimal frame size should be set as n, such that the probability of each slot carrying a single ID (without collision) can be maximized. This can be easily seen as follows: Consider an arbitrary slot. The probability p that one and only one tag chooses this slot to transmit is

$$p = \binom{n}{1}\frac{1}{f}\left(1-\frac{1}{f}\right)^{n-1} \approx \frac{n}{f}e^{-\frac{n-1}{f}} \approx \frac{n}{f}e^{-\frac{n}{f}} \qquad (2)$$

when n is large. To find the value of f that maximizes p, we take the first-order derivative of the right side and set it to zero. Solving the resulting equation, we have

$$f = n, \qquad (3)$$

which means the maximal value of p is e^{-1}. In subsequent requests, as more and more IDs have been collected, fewer and fewer tags are transmitting their IDs and the frame sizes should be reduced accordingly.

However, we do not know n. There are numerous estimation methods for n [14]–[16], which are however intended for a system with a large number of tags, in tens of thousands. It is known that these estimation methods will actually be inefficient if they are applied to a relatively small number of tags such as a couple of thousands or fewer [17]; if the number of tags is very small, the estimation time can be much larger than the time it takes to complete the tag identification task

itself. In the context of this paper, we expect the number of children of the reader or any tag is relatively small. Hence, it is not worthwhile to add the overhead of a separate component for estimating n before the reader (tag) begins collecting IDs from its neighborhood.

Our solution is to estimate the value of n iteratively from the frame itself without incurring additional overhead. Initially, we set f to be a small constant λ in the first request. We double the value of f in each subsequent request until there exists at least one empty slot that no tag chooses. From then on, we will estimate the number of n and set the frame size accordingly in the subsequent requests. Without the loss of generality, suppose we want to determine the frame size for the ith request. Let f_j be the frame size used in the jth request, $1 \leq j < i$. After the jth request, let c_j, s_j, and e_j be the numbers of slots that are chosen by multiple tags (collision), a single tag, and zero tag, respectively. Let m_j be the number of IDs that are successively collected after the jth request. All these values are known to the reader (tag). The process for a tag to randomly choose a slot in a time frame can be cast into bins and balls problem [18]. In the jth frame, $n - m_{j-1}$ tags (balls) are mapped to f_j slots (bins). The total number of different ways for putting $n - m_{j-1}$ balls to f_j bins is ${f_j}^{n-m_{j-1}}$. The number of ways for choosing e_j bins from f_j bins and let them be empty is $\binom{f_j}{e_j}$. In addition, the number of ways for choosing s_j balls from $n-m_{j-1}$ balls and putting each of them into one of the remaining $f_j - e_j$ bins is $\binom{f_j-e_j}{s_j}\binom{n-m_{j-1}}{s_j}(s_j!)$. Finally, the remaining $n - m_{j-1} - s_j$ balls should be thrown into the remaining c_j bins, each containing at least 2 balls (collision slots). We first choose $2c_j$ balls and put 2 balls into each of the c_j bins, which includes $\binom{n-m_{j-1}-s_j}{2c_j}\frac{(2c_j)!}{2^{c_j}}$ possibilities. After that, the remaining $(n-m_{j-1}-s_j-2c_j)$ balls can be put into any of the c_j bins, which involves $(n - m_{j-1} - s_j - 2c_j)^{c_j}$ different ways. Therefore, the likelihood function for observing these values is

$$L(n) = \prod_{j=1}^{i-1} \frac{\binom{f_j}{e_j}\binom{f_j-e_j}{s_j}\binom{n-m_{j-1}}{s_j}(s_j!)\binom{n-m_{j-1}-s_j}{2c_j}\frac{(2c_j)!}{2^{c_j}}}{{f_j}^{n-m_{j-1}}} \times (n - m_{j-1} - s_j - 2c_j)^{c_j}. \quad (4)$$

The estimate of n is the value that maximizes L. Let this value be $\hat{n}$, which can be found through exhaustive search since the range for n is limited in practice, rarely going beyond tens of thousands. For the ith request, we set the frame size to be $\hat{n} - m_{i-1}$.

The above estimator follows the general principle originally seen in [14], but it takes the information of c_j, s_j, and e_j all in the same estimator, whereas the estimators in [14] use either c_j or e_j.

As our analysis will show, except for the reader, the average number of children per tag is typically very small (less than 2) for a randomly distributed tag network. In this case, if we set the initial frame size λ to 4, the chance is high that a tag successfully collect all IDs from it children in the first time frame. Therefore, only the reader needs to use (4) to estimate the number of its children, while the tags can just set the frame size to a small constant to avoid the computation overhead.

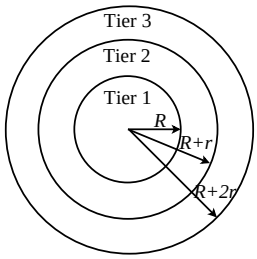

Fig. 7: An illustration of a network with three tiers of tags.

i	1	2	3	4	5	6	7	8	9
D_i	1.3	1.2	1.2	1.2	1.1	1.1	1.1	1.1	1.1

TABLE I: The values of D_i with $R = 3r$

I. Load Factor Per Tag

We analyze the work load of each tag in terms of how many children and descendants it has to deal with. While our load balancing approach is designed for any tag distribution, to make the analysis tractable, we assume here that tags are evenly distributed in an area with density ρ, and the tags whose distances from the reader are no larger than R form the first tier, while those whose distances from the reader are greater than $R + (i-2)r$ but smaller than $R + (i-1)r$ form the ith ($i \geq 2$) tier of the network, where the transmission ranges of the reader and a tag are R and r, respectively, with $R \geq r$. For example, Fig. 7 presents a network with three tiers. The number N_i of tags in the ith tier is estimated as

$$\begin{aligned} N_i &= \rho \times (\pi \times (R + (i-1)r)^2 - \pi \times (R + (i-2)r)^2) \\ &= \pi\rho(2Rr + (2i-1)r^2). \end{aligned} \quad (5)$$

One exception is that N_1 computed from (5) actually includes only the portion of tier-1 tags whose distances from the reader are larger than $R-r$; these are the tags that can serve as parents for tier-2 tags.

The *children degree* of tier-i tags, denoted by D_i, is defined as the average number of children that a tier-i tag has. Because tags at the ith tier only serve as parents for tags at the $(i+1)$th tier, we have

$$D_i = \frac{N_{i+1}}{N_i} = \frac{2R + (2i+1)r}{2R + (2i-1)r} = 1 + \frac{1}{\frac{R}{r} + (i - \frac{1}{2})}. \quad (6)$$

We have $R \gg r$ because the reader can transmit at a much higher power level and it has much more sensitive antenna. This makes the values of D_i very small. For example, if $R = 3r$, Table I shows the values of D_i, $1 \leq i < 10$, which are smaller than 1.3 and quickly converge toward 1 as i increases. The values in the table will be even smaller if $R > 3r$.

i	1	2	3	4	5	6	7	8	9
L_i	21.9	16.0	12.1	9.2	7.0	5.2	3.6	2.3	1.1

TABLE II: The values of L_i with $R = 3r$ and $l = 10$

The *load factor* of tier-i tags, denoted as L_i, is defined as the average number of IDs that a tier-i tag has to forward, including the IDs of its tier-$(i+1)$ children as well as other IDs that its children collects from their descendants. L_i is equal to the total number of tags beyond the ith tier divided by the number of tags at the ith tier.

$$\begin{aligned} L_i &= \frac{\sum_{j=i+1}^{l} N_j}{N_i} = \frac{\sum_{j=i+1}^{l} 2R + (2j-1)r}{2R + (2i-1)r} \\ &= \frac{2(l-i) + \frac{r}{R}(l^2 - i^2)}{2 + (2i-1)\frac{r}{R}}, \end{aligned} \tag{7}$$

where l is the total number of tiers and $i < l$. When $R = 3r$ and $l = 10$, Table II shows the values of L_i, $1 \leq i < 10$, which are surprisingly small. Because tier-1 tags can be powered by the radio wave from the reader, we are only concerned with the power consumption of tags at other tiers. The tags at tier 2 have to forward more IDs than those at outer tiers. From the table, a tier-2 tag forwards just 16 IDs on average, which is modest overhead, considering that there are 8 more tiers beyond tier 2.

While the average is modest, the worst-case load factor is also important when we evaluate overhead. SICP is designed to evenly distribute the work load among tags by balancing the spanning tree, so that tags at a certain tier have similar numbers of children (or descendants), which translate to similar children degrees (or load factors). We will study the worst-case children degree and load factor by simulations.

V. Evaluation

A. Simulation setup

There is no prior work on tag identification for networked tag systems[1]. But known techniques such as broadcast and contention-based transmission widely used in other wireless systems can be used to design a state-free tag identification protocol, CICP, which we will use as a benchmark for comparison. We evaluate the performance of CICP and SICP to demonstrate two major findings that (1) although the ALOHA-based protocols are very successful in other wireless systems (including RFID systems), they are not suitable for networked tag systems, and that (2) serialization can significantly improve the tag identification performance.

Three performance metrics are used: (1) execution time measured in number of time slots, (2) average and maximum numbers of bits sent per tag, and (3) average and maximum numbers of bits received per tag. The last two are indirect measures of energy cost, where tier-1 tags are excluded because they can be powered by the reader's radio waves. Computation by tags in the proposed protocols is very limited. Most energy is spent on communication. The amount of communication data serves as an indirect means to compare different protocols. For example, if tags in one protocol receive and send *far more* than those in another protocol, it is safe to say that the first protocol costs more energy than the second.

We vary the number N of tags in the system from 1000 to 10000 at steps of 1000. The tags are randomly distributed in a circular area with a radius of 50 m. The reader, whose communication range R is set to 25 m, is located at the center of the area. For each tag, its inter-tag communication range r is 5 m. In SICP, the reader sets its frame size of the ith request to $f_i = \max\{\hat{n} - m_{i-1}, f_l\}$, where m_{i-1} is the number of IDs that have been collected and $\hat{n}$ is the estimate number of tags that maximizes (4). The lower bound f_l, fixed to 50, prevents the frame size from being setting too small or even negative due to the estimation deviation of $\hat{n}$. The initial frame size λ is 50 for the reader. To relieve the tags from estimating the numbers of children they have, we let them use a fixed frame size λ with a default value of 4, but we will also vary it from 2 to 10. The length of each tag ID is 96 bits long. The length of each serial number is $\lceil \log_2 N \rceil$ bits long. The length of each tier number is 4 bits long. Following the specification of the EPC global Class-1 Gen-2 standard [19], we set the length of the ID collection request in CICP to 20 bits, and set ACK and NAK to 16 bits and 8 bits, respectively. In SICP, the ACK will also include a serial number. For each data point in the figures, we repeat the simulation for 100 times and present the average result.

B. Children Degree and Load Factor

We first examine the balance of the spanning trees built by CICP and SICP. It has significant impact on the worst-case energy cost of the tags. A tag with a larger children degree (or a larger load factor) has to collect (or forward) more tag IDs, resulting in additional energy expenditure. Tags that have the largest children degree or load factor may become the energy bottleneck in the network. If the residual on-tag energy is exhausted before the completion of the protocol, the network may even be partitioned due to dead tags.

Fig. 8 and Fig. 9 present the maximum children degree and the maximum load factor in the spanning trees built by CICP and SICP, respectively. As the number N of tags in the system becomes larger, the increase in these worst-case numbers under CICP is a lot faster than the increase under SICP, indicating a much balanced tree for the latter. For example, when $N = 10000$, the maximum children degree and load factor in CICP are 83 and 1969, and those numbers in SICP are only 12 and 259.

C. Performance Comparison

We compare the performance of CICP and SICP in Fig. 10, where the first plot shows the protocol execution time in terms of number of slots used, the second plot shows the average number of bits sent per tag, and the third plot shows the average number of bits received per tag. SICP uses slightly more slots than CICP, meaning that its execution time is modestly longer. However, its energy cost is much smaller, thanks to serialization for collision reduction. For example,

[1]For the special case when all networked tags are within the direct coverage of the reader, our protocols naturally become the traditional protocols, literally, because we may actually adopt any existing ALOHA-based RFID identification protocol for collecting IDs within the reader's neighborhood in place of the operations described in Section IV-D, as long as the serial number is embedded in ACK.

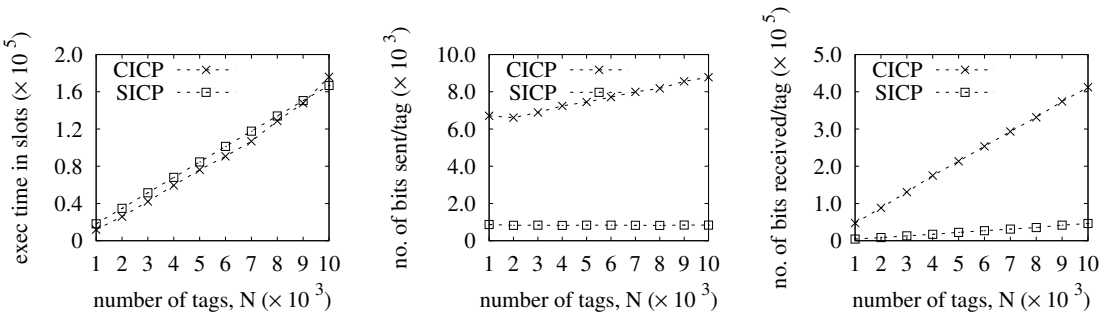

Fig. 10: Performance comparison between CICP and SICP.

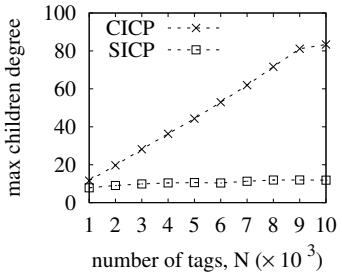

Fig. 8: Maximum children degree in the spanning tree.

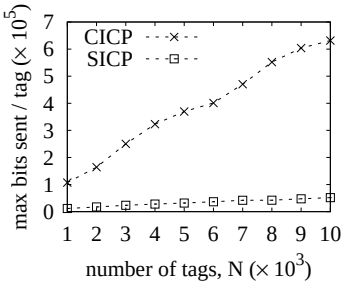

Fig. 11: Maximum number of bits sent by any tag.

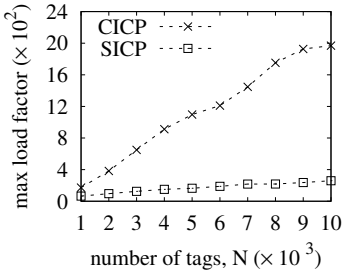

Fig. 9: Maximum load factor of tags in the spanning tree.

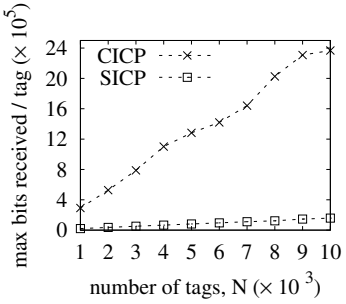

Fig. 12: Maximum number of bits received by any tag.

when $N = 10000$, the numbers of bits sent/received per tag in CICP are 8783 and 412218, whereas those numbers are just 839 and 46217 — 90.0% and 88.8% reduction, respectively.

Fig. 11 and Fig. 12 show the maximum numbers of bits sent/received by a tag under the two protocols, respectively. As expected, the most energy-consuming tags spend much less energy under SICP than under CICP. For example, when $N = 10000$, the maximum numbers of bits sent/received by any tag in CICP are 631412 and 2367899, and those numbers in SICP are 51787 and 158458 — 91.8% and 93.3% reduction, respectively.

D. Performance Tradeoff for SICP

Finally, we demonstrate a performance tradeoff for SICP controlled by the value of λ. We set $N = 5000$ and vary λ from 2 to 10. The results are presented in Fig. 13, where the three plots from left to right show the execution time, the average number of bits sent per tag, and the average number of bits received per tag, respectively. As the value of λ increases, the execution time increases, but the energy cost for sending and receiving decreases. This presents a time-energy tradeoff. However, the time increases almost linearly, but the decrease in energy flattens out, suggesting that a modest value of λ is preferred.

VI. Related Work

The tag identification protocols for traditional RFID systems can be broadly classified into two categories: *ALOHA-based* [12], [13], and *tree-based* [10], [11]. To run an ALOHA-based identification protocol, the reader first broadcasts a query, which is followed by a slotted time frame. Each tag randomly picks a time slot in the frame to report its ID. Collision happens if a slot is chosen by multiple tags. Tags not receiving positive acknowledgements from the reader will continue participating in the subsequent frames. The dynamic frame slotted ALOHA (DFSA) [20], [21] adjusts the frame

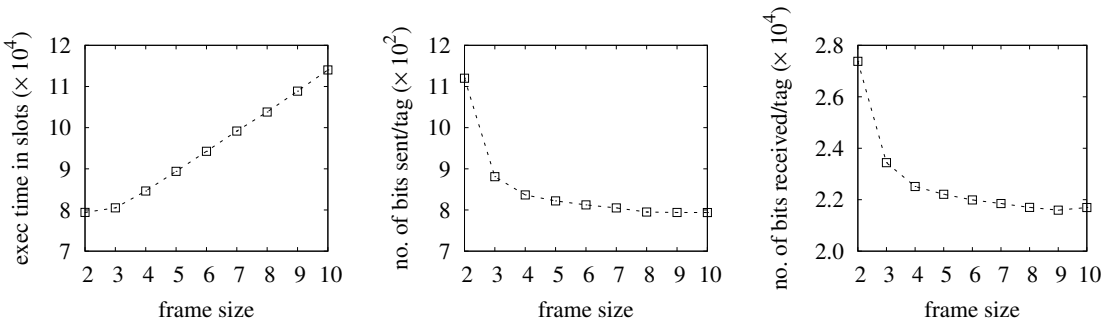

Fig. 13: Execution time and energy cost of SICP with respect to λ, when $N = 5000$.

size round by round.

The tree-based protocols organize all IDs into a tree of ID prefixes. Each in-tree node has two child nodes that have one additional bit, '0' or '1'. The tag IDs are leaves of the tree. The 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.

To further improve the identification efficiency, network coding and interference cancelation techniques are used to help the reader recover IDs from collided signals [22], [23].

VII. Conclusion

This paper is the first study on tag identification in the emerging networked tag systems. The multihop nature of networked tag systems makes this problem different from the tag identification problem in RFID systems. We propose two tag identification protocols with two important findings. The first finding is that the traditional contention-based protocol design incurs too much energy overhead in networked tag systems due to excessive collision. The second finding is that load imbalance causes large worst-case energy cost to the tags. We address these problems through serialization and probabilistic parent selection based on serial numbers.

VIII. Acknowledgments

This work was supported in part by the National Science Foundation under grant NeTS-1409797.

References

[1] Y. Qiao, S. Chen, and T. Li, "Energy-efficient Polling Protocols in RFID Systems," *Proc. of ACM Mobihoc*, May 2011.

[2] S. Chen, M. Zhang, and B. Xiao, "Efficient Information Collection Protocols for Sensor-augmented RFID Networks," *Proc. of IEEE INFOCOM*, pp. 3101–3109, April 2011.

[3] M. Chen, W. Luo, Z. Mo, S. Chen, and Y. Fang, "An Efficient Tag Search Protocol in Large-Scale RFID Systems," *Proc. of IEEE INFOCOM*, pp. 1325–1333, April 2013.

[4] J. Liu, B. Xiao, S. Chen, F. Zhu, and L. Chen, "Fast rfid grouping protocols," pp. 1948–1956, 2015.

[5] K. Bu, B. Xiao, Q. Xiao, and S. Chen, "Efficient Pinpointing of Misplaced Tags in Large RFID Systems," *Proc. of IEEE SECON*, pp. 287–295, 2011.

[6] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman, "Challenge: Ultra-Low-Power Energy-Harvesting Active Networked Tags (EnHANTs)," *Proc. of ACM Mobicom*, pp. 253–260, September 2009.

[7] M. Gorlatova, R. Margolies, J. Sarik, G. Stanje, J. Zhu, B. Vigraham, M. Szczodrak, L. Carloni, P. Kinget, I. Kymissis, and G. Zussman, "Prototyping Energy Harvesting Active Networked Tags (EnHANTs)," *Proc. IEEE INFOCOM mini-conference*, April 2013.

[8] P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman, "Energy Harvesting Active Networked Tags (EnHANTs) for Ubiquitous Object Networking," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 18–25, December 2010.

[9] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient Backscatter: Wireless Communication Out of Thin Air," *Proc. of ACM SIGCOMM*, pp. 39 – 50, August 2013.

[10] M. Shahzad and A. Liu, "Probabilistic Optimal Tree Hopping for RFID Identification," *Proc. of ACM SIGMETRICS*, pp. 293–304, June 2013.

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

[12] Q. Chen, Y. Liu, H. Ngan, and L. M. Ni, "ASAP: Scalable Identification and Counting for Contactless RFID Systems," *Proc. of IEEE ICDCS*, 2010.

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

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

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

[16] M. Shahzad and A. Liu, "Every Bit Counts - Fast and Scalable RFID Estimation," *Proc. of ACM MOBICOM*, August 2012.

[17] W. Luo, Y. Qiao, and S. Chen, "An Efficient Protocol for RFID Multigroup Threshold-based Classification," *Proc. of IEEE INFOCOM*, 2013.

[18] N. L. Johnson and S. Kotz, *Urn Models and Their Application: An Approach to Modern Discrete Probability Theory*. John Wiley and Sons Inc, 1977.

[19] "EPC Radio-Frequency Identity Protocols Class-1 Gen-2 UHF RFID Protocol for Communications at 860MHz-960MHz, EPCglobal," *http://www.epcglobalinc.org/uhfclg2*, April 2011.

[20] C. T. Nguyen, K. Hayashi, M. Kaneko, P. Popovski, and H. Sakai, "Probabilistic Dynamic Framed Slotted ALOHA for RFID Tag Identification," *Wireless Personal Communications*, vol. 71, pp. 2947–2963, August 2013.

[21] I. Onat and A. Miri, "A Tag Count Estimation Algorithm for Dynamic Framed ALOHA Based RFID MAC Protocols," *Proc. of IEEE ICC*, June 2011.

[22] L. Kong, L. He, Y. Gu, M. Wu, and T. He, "A Parallel Identification Protocol for RFID Systems," *Proc. of IEEE INFOCOM*, pp. 154 – 162, April-May 2014.

[23] M. Zhang, T. Li, S. Chen, and B. Li, "Using Analog Network Coding to Improve the RFID Reading Throughput," *Proc. of IEEE ICDCS*, 2010.