Efficient and reliable low-power backscatter networks

Efficient and Reliable Low-Power Backscatter Networks

Jue Wang Haitham Hassanieh Dina Katabi Piotr Indyk
Massachusetts Institute of Technology
{jue_w, haithamh, dk, indyk}@mit.edu

Abstract – There is a long-standing vision of embedding backscatter nodes like RFIDs into everyday objects to build ultra-low power ubiquitous networks. A major problem that has challenged this vision is that backscatter communication is neither reliable nor efficient. Backscatter nodes cannot sense each other, and hence tend to suffer from colliding transmissions. Further, they are ineffective at adapting the bit rate to channel conditions, and thus miss opportunities to increase throughput, or transmit above capacity causing errors.

This paper introduces a new approach to backscatter communication. The key idea is to treat all nodes as if they were a single virtual sender. One can then view collisions as a code across the bits transmitted by the nodes. By ensuring only a few nodes collide at any time, we make collisions act as a *sparse code* and decode them using a new customized compressive sensing algorithm. Further, we can make these collisions act as a *rateless code* to automatically adapt the bit rate to channel quality –i.e., nodes can keep colliding until the base station has collected enough collisions to decode. Results from a network of backscatter nodes communicating with a USRP backscatter base station demonstrate that the new design produces a 3.5× throughput gain, and due to its rateless code, reduces message loss rate in challenging scenarios from 50% to zero.

Categories and Subject Descriptors C.2.2 [**Computer Systems Organization**]: Computer-Communications Networks

General Terms Algorithms, Design, Performance

Keywords Backscatter, RFID, Wireless, Compressive Sensing

1. INTRODUCTION

Backscatter devices – like RFIDs – differ from other low-power communication technologies (UWB, ZigBee, Bluetooth, etc.) in that they do not expend their own energy on data transmission [16]. In backscatter systems, a device called *the reader* transmits a high power RF signal. A nearby backscatter node can modulate the reader's signal by changing the impedance match on its own antenna in order to convey a message of zeros and ones back to the reader. This behavior allows a node to communicate at almost zero power, by capturing the energy in the reader's RF signal [24]. Further, new generations of backscatter nodes are equipped with various sensing capabilities and can communicate over multiple meters [52]. They may also come equipped with a small onboard battery for sensing and computation but use backscatter to reduce communication cost [42]. These capabilities make them ideal for ultra-low power ubiquitous networks. In fact, there is a growing interest in deploying both RFID and integrated RFID-sensor networks [19, 45]. A typical application involves tagging all items in a store with RFIDs, so that a customer can pay for her purchases simply by pushing her shopping cart through the checkout line [6]. Other applications include monitoring temperature, pressure, humidity, etc. [18, 45]. Recent reports show that industries such as healthcare, retailing, oil and aviation are moving towards deploying such backscatter networks for object tracking, asset monitoring, and emerging Machine-to-Machine (M2M) applications [18, 19, 38].

There are two main challenges, however, in deploying reliable and efficient backscatter networks:

- *(a) Node Identification Overhead:* Backscatter nodes cannot hear each other's transmissions [8]. Thus, they rely on the reader to schedule their medium access. The reader, however, does not know a priori which nodes want to transmit (e.g., which items are in a customer's shopping cart). Thus, existing backscatter protocols [14] go through an identification phase before transmission, which consumes a significant fraction of the total communication time (30%-60% in common modes of RFID EPC Gen-2 standard [14]). At a high level, the reader divides time into slots. A node that wants to transmit picks a random temporary id and sends it in a random slot. Since the nodes cannot hear each other, some of them may collide. To increase reliability one needs to eliminate collisions. Eliminating collisions however requires making the total number of time slots fairly large, which reduces efficiency as many empty slots are produced.
- *(b) Ineffective Bit Rate Adaptation:* Ultra-low power technologies, including backscatter, use very simple modulations that transmit only 1 bit/symbol (e.g., ON-OFF keying or BPSK) [48, 16]. Denser modulation schemes (e.g., 4QAM, 16QAM, or 64QAM) are avoided because they require power hungry linear amplifiers [35]. This leaves backscatter nodes unable to leverage a good channel to transmit more bits per symbol and increase bandwidth efficiency. It is also difficult for them to reduce the bit rate to increase robustness to bad channels [7]. Such adaptation requires the reader to send feedback to each node so it can adjust the redundancy in its messages. However, such feedback is not cost effective because backscatter messages are short and often bursty [14].

This paper introduces Buzz, a communication system for backscatter networks that improves both reliability and efficiency. The key idea underlying Buzz is to treat backscatter nodes as if they were a single virtual sender. Collisions can then be viewed as a code across the bits transmitted by different nodes. However, simply letting nodes repeatedly collide leads to multiple copies of the same collision (i.e., the same codeword), which is not effective for decoding [46]. Instead, Buzz makes each node transmit only in a small random subset of the collisions, and does so without imposing overhead on the nodes themselves. The resulting code is sparse and also rateless (i.e., nodes can keep colliding until the reader collects enough collisions to decode). Buzz then leverages the sparsity

SIGCOMM'12, August 13–17, 2012, Helsinki, Finland.

and rateless nature of the code to enable fast identification and distributed rate adaptation, as described below.

Node Identification: We would like to identify the K nodes that want to transmit in a network of N nodes, where $K \ll N$ (e.g., 20 items in a customer's shopping cart among one million items in a Wal-Mart store). We can model the scenario as an N-element *sparse* binary vector $\mathbf{x}$ that is zero everywhere except in K locations. Let all K nodes that have data transmit concurrently, where a node i transmits a known binary pattern A_i. The signal received by the reader, $\mathbf{y}$, can be represented as:

$$\begin{aligned} \mathbf{y} &= [A_1 \ldots A_i \ldots A_N]\mathbf{x} \\ &= \mathbf{A}\mathbf{x} \end{aligned} \quad (1)$$

Eq. 1 is a standard compressive sensing problem, where one wants to retrieve a *sparse* vector $\mathbf{x}$ using linear combinations $\mathbf{y} = \mathbf{A}\mathbf{x}$. Thus, we can use a compressive sensing solution to efficiently recover $\mathbf{x}$ with a small number of linear combinations.

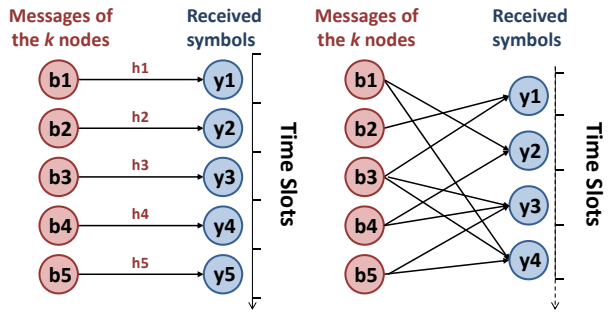
However, traditional compressive sensing algorithms are computationally infeasible to apply in this setting because $\mathbf{A}$ has as many columns as the number of nodes in the network (e.g., one million items in a Wal-Mart store). To develop a practical solution, we exploit the sparsity in $\mathbf{x}$ to eliminate large chunks of columns in $\mathbf{A}$. In particular, we hash the elements of $\mathbf{x}$ into buckets. All ids that hash to empty buckets can be eliminated while ids that hash to non-empty buckets can be disambiguated using a much smaller scale compressive sensing solution. In §5, we incorporate the effect of the wireless channel and extend this idea to form a full-fledged identification protocol for backscatter networks.

Distributed Bit Rate Adaption: In contrast to traditional rate adaptation, which focuses on point-to-point communication, Buzz looks at the network as a whole and adapts the aggregate bit rate of all backscatter nodes to channel conditions. Fig. 1(a) shows the existing system where backscatter nodes transmit sequentially. In this design, each node's share of the medium is the same. A node with a good channel probably does not need the amount of share it gets, while another node with a bad channel would not be able to deliver its data within its share. In contrast, backscatter nodes in Buzz randomly collide in different time slots and keep doing so until the reader signals that it has correctly received their data, as shown in Fig. 1(b). These collisions act as a rateless code *across* nodes in the network and allow us to implicitly redistribute the mismatched network resources.

However, decoding these collisions to recover the transmitted bits is an expensive joint optimization problem [51]. To address this issue, in Buzz each node contributes to only a small random subset of the collisions so that the resulting code is sparse, i.e., has a low density. Similar to LDPC codes, such low density codes can be decoded using a linear time decoder based on belief propagation. However, in contrast to LDPC codes, which are centralized block codes, Buzz's code is both distributed (i.e., it operates across the bits of many nodes) and rateless (i.e., the reader collects collisions until it has enough to decode). Using these properties, Buzz provides automatic bit rate adaptation to backscatter networks.

Summary of Results: We built a prototype of Buzz and evaluated it in a testbed of 16 computational UHF RFIDs and a USRP backscatter reader. We compared Buzz with TDMA and CDMA based schemes in a wide range of channel conditions, which leads to the following findings:

- Averaged across experiments with different numbers of concurrent tags and over 600 traces in different channel conditions, Buzz improved the overall communication efficiency of backscatter networks by 3.5×. This gain is the combination of two factors: First, Buzz reduced the time spent in the identification phase by 5.5×, compared to the Framed Slotted Aloha protocol used in the EPC Gen-2 standard; Second, during the data transmission phase, Buzz's bit rate adaptation on average delivered a throughput gain of 2× over TDMA and CDMA.
- Buzz enables backscatter networks to work in far more challenging channel conditions than previously possible. In challenging conditions, TDMA and CDMA systems experienced a message loss rate as high as 50% and 100% respectively whereas Buzz's loss rate was zero due to its rateless code.



(a) Avoiding Collisions (b) Allowing Collisions

Figure 1—(a) Current systems try to avoid collisions by having nodes sequentially transmit. (b) In Buzz a random subset of nodes collide in each time slot, forming a distributed rateless code.

Contributions: This paper makes the following contributions:

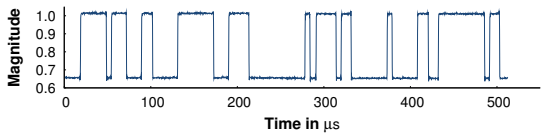
- It introduces the concept of using randomized collisions as a distributed code across the bits of multiple transmitters.
- It presents a new low-complexity compressive sensing algorithm that enables fast backscatter node identification.
- It presents the first automatic rate adaptation protocol that adapts the collective bit rate of multiple transmitters, which do not individually change their transmission bit rate.
- It demonstrates a working system that provides a severalfold improvement to the efficiency and reliability of backscatter networks.

2. BACKSCATTER COMMUNICATION

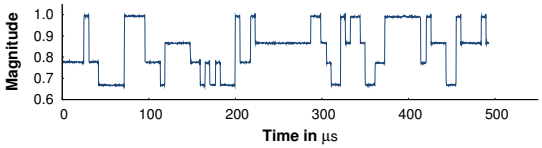
In backscatter networks, the reader transmits a high power continuous waveform. Backscatter nodes transmit their signal by reflecting back the continuous waveform using ON-OFF keying. The nodes transmit a "1" bit by changing the impedance on their antennas to reflect the reader's signal and a "0" bit by remaining in their initial silent state [16].

There are four main distinctions between backscatter networks and the more familiar WiFi networks.

- There is no carrier frequency offset Δf_c between different nodes' transmissions since nodes do not generate their own RF signal but rather reflect the reader's signal [16].
- Backscatter nodes transmit and receive in a narrow bandwidth due to their power limitation [14]. As a result, the multipath effect of wireless communication is negligible and the system can be modeled as a single tap channel (one complex number) [46].
- Nodes are naturally synchronized by the reader's query and small synchronization errors do not matter since they transmit at very low bit rates (tens to hundreds of kbps) [14]. In §8.1, we present measurement results for commercial passive RFID tags and computational RFID tags.

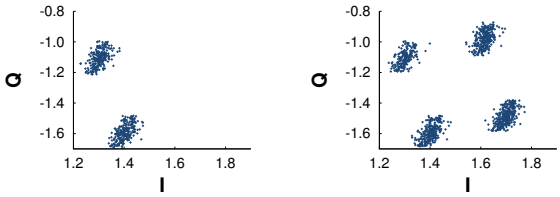


(a) A Single Node Transmission



(b) Collision of Two Nodes' Transmissions

Figure 2—(a) When a single node transmits, the received signal has two levels which distinguish a "0" bit and a "1" bit. (b) When two nodes' signals collide, the received signal has four levels corresponding to "00", "01", "10" and "11".



(a) Single Node Constellation (b) Two Node Constellation

Figure 3—(a) A single node's transmission has two constellation points. (b) Concurrent transmissions of two nodes lead to four constellation points. Decoding a denser constellation increases the throughput of the network.

- The reader is a single power-full device and decoding complexity can be delegated to it while keeping the backscatter nodes simple and power efficient [27].

3. ILLUSTRATIVE EXAMPLES

We start with two simple examples that illustrate the benefits of collisions and provide some intuition into how one may harvest these opportunities.

3.1 Collisions Enable Distributed Rate Adaptation

As mentioned in §1, ultra-low power devices typically use simple modulations that transmit only one bit per symbol (e.g., ON-OFF keying, BPSK, or binary ASK) [48]. Such a choice of modulation is conservative, i.e., it assumes bad channel conditions. In many cases, the wireless channel can support denser modulations that transmit multiple bits per symbol, e.g., 4QAM, 16QAM or higher. Today's low power devices cannot harvest these opportunities to transmit at higher bit rates and increase their throughput.

Collisions can address the issue. Fig. 2 shows backscatter transmission signals received at a USRP-based reader. When a single node transmits (Fig 2(a)), the signal exhibits two levels, which can be used to distinguish a "1" bit from a "0" bit. In contrast, when transmissions from two backscatter nodes collide (Fig. 2(b)), the received signal exhibits four levels that correspond to the values of the two colliding bits: "00", "01", "10", and "11". Knowing the channel coefficients of the nodes, a backscatter reader that receives the signal in Fig. 2(b) can easily distinguish these four levels, and decode both nodes from a single collision, obtaining a bit rate of 2 bits/symbol. Hence, by letting these two nodes collide, one can double the aggregate bit rate.

In fact, letting two backscatter nodes collide is fairly similar

TX	Slot 1	Slot 2	Slot 3
Pattern 1	0	1	1
Pattern 2	1	0	0
Pattern 3	1	0	1
Pattern 4	1	1	1

Table 1—Transmit Patterns

Pattern	011	100	101	111
011	~~022~~	111	112	122
100	111	~~200~~	201	211
101	112	201	~~202~~	212
111	122	211	212	~~222~~

Table 2—Collision Patterns

to having one virtual node transmit using a denser modulation. This can be more easily seen by comparing Fig. 3(a), which plots the constellation of one backscatter node transmitting alone, and Fig. 3(b), which plots the denser constellation with two nodes transmitting together. Allowing two nodes to collide produces a 4-point constellation. The difference between the two constellations in Fig. 3 resembles the difference between BPSK, a 1 bit/symbol modulation, and 4QAM, a 2 bits/symbol modulation. Allowing more nodes to collide leads to even denser constellations (e.g., three colliding nodes produce an 8-point constellation).

The above example reveals an opportunity for increasing network throughput via collisions. Harvesting the opportunity however requires more than simply letting nodes collide. Decoding is possible in Fig. 3(b) because the four constellation points are distinguishable. If the channels however were noisier or the spacing of the constellation were less ideal, the different levels in Fig. 2(b) could be confused, leading to decoding errors. Without an additional mechanism, the backscatter nodes would need to know the channel's signal-to-noise ratio (SNR) a priori to decide whether it can support 2 bits/symbol. To address this issue, we design a rateless code to enable automatic rate adaptation. In contrast to conventional rateless codes [25, 39], the new code operates across multiple nodes to adapt their aggregate bit rate. The encoding at each node involves only transmitting its original message, or remaining silent. In §6, we describe the code and its corresponding decoder.

3.2 Collisions Facilitate Assigning Unique IDs

It might seem that collisions make it harder to distinguish different nodes. However, in this section we will show that designing for collisions can improve the system's ability to assign unique temporary ids to the nodes. Consider the following toy example, where two backscatter nodes need to obtain unique ids. The reader divides time into slots and the two nodes transmit in these slots. Suppose that the total time is three slots. We will compare the probabilities that the two nodes will obtain unique ids in two designs:

- **Option 1:** Each node picks a time slot at random and transmits in it. If the two nodes pick different time slots, they can use their time slots as their unique ids.
- **Option 2:** Each node transmits in all three time slots, but the transmission pattern is randomly chosen from the following set of patterns: 011, 100, 101 and 111. It then transmits that pattern over the period of the three time slots, as shown in Table 1. In this case, the pattern serves as the node's id.

In option 1, the two nodes will fail to obtain unique ids if they transmit in the same time slot , which occurs with probability 1/3. In option 2, the nodes will collide in the three slots. For simplicity, let us assume that the channels of the two nodes are the same. The reader will receive one of the collision patterns in Table 2. The reader can then map the received pattern to identify which two patterns were picked by the two nodes. Since there are four possible patterns, the two nodes will be indistinguishable with probability 1/4. Note that while the two options have the same overhead (i.e., three time slots), option 2 reduces the probability that the two nodes end up with indistinguishable ids from 1/3 to 1/4. This toy example shows that designing for collisions can improve the distinguishability of the nodes and reduce the probability of failing to assign them

different ids. In §5 we propose a new compressive sensing algorithm which generalizes this basic intuition to any number of nodes and practical settings.

4. PROBLEM DOMAIN

Buzz is designed to increase communication efficiency and reliability in backscatter networks. We consider a deployment of backscatter nodes that capture some measurements of interest (e.g., temperature readings, object tracking). The setup is fairly similar to that of a wireless LAN where a backscatter reader plays the role of the access point. Both uplink and downlink communications have to go through the reader.

Before we delve into the details of our design, it is important to distinguish two modes of operation for backscatter networks:

(a) Event-Driven Communication: This is the more common mode of operation where a backscatter network is used to detect certain events of interest. The classic example is the shopping cart application, in which backscatter nodes are attached to items in a store [6]. A customer pays for her purchases simply by pushing her cart past the scanning area, where a backscatter reader queries the items in the cart for their ids and reports the ids to an application that looks up their prices and presents the charge to the customer. Other event-driven applications include inventory management, object tracking, and event detection [18, 38].

Communication in event-driven networks can be divided into three phases. We describe them at a high level and refer the reader to the EPC Gen-2 standard for more details:

- *Identification:* First the reader has to identify the nodes that have data to transmit. The identification phase uses temporary ids. This is because the *globally* unique id is often long, while the *temporary* id can be short since the uniqueness needs to hold only for the nodes that want to transmit around the same time. For example, the EPC Gen-2 standard uses 16-bit temporary ids during the identification phase [14].
- *Resource Allocation:* In the next step the reader schedules node transmissions. In the EPC Gen-2 standard, the reader assigns the nodes different time slots [14]. In a CDMA or FDMA system, the reader assigns the nodes different CDMA codes or different frequency bands. In these schemes, the reader has to transmit to each node its corresponding assignment.
- *Uplink Data Transmission:* Finally, the nodes transmit their data to the reader using the allocation in the previous phase. The data could be the node's global id or other context information.

Event-driven networks can use Buzz as follows: To assign unique temporary ids, they can use Buzz's compressive sensing identification protocol. As for the second phase, Buzz eliminates the need for the reader to allocate resources to the nodes because it enables the nodes to use a randomized approach to access the medium and transmit their data. Hence, in Buzz the nodes can immediately proceed to the third phase and transmit their data using Buzz's rate adaptation protocol.

(b) Periodic Backscatter Networks: Here, backscatter nodes sense some process and periodically report their measurements to the reader. An example application of such networks is creating real-time heat maps of a data center, where low-power backscatter nodes periodically sense and report the temperature [45].

In periodic backscatter networks, the set of nodes that have data is known a priori. Hence, the network can adopt a static schedule for accessing the medium which eliminates the need for an identification phase. Therefore, periodic backscatter networks can directly use Buzz's rateless bit rate adaptation protocol to maximize the efficiency and reliability of their communication.

5. NODE IDENTIFICATION USING COMPRESSIVE SENSING

We have a backscatter network with N nodes, but only $K \ll N$ have data. We would like to identify these K nodes. We can model this problem as follows: consider a binary vector $\mathbf{x}_{N\times 1}$ where each element x_i corresponds to the backscatter node with id i and $x_i = 1$ if node i has data to transmit. Our aim is to estimate $\mathbf{x}$. One option for solving the problem is to allocate N time slots that correspond to the N elements of $\mathbf{x}$, and have each of the K nodes transmit in its own time slot (i.e., send a "1" bit). This option addresses the collision problem since each node sends in a unique time slot, but it is very wasteful because $K \ll N$.

Protocol: In our scheme, the reader triggers the nodes to start transmitting. Each node that has data (i.e., $x_i = 1$) then uses its id i as a seed to a pseudorandom binary number generator.[1] For each time slot, the node generates a random bit ("0" or "1") and transmits if the random bit is "1". Therefore in each time slot, the reader receives one wireless symbol, which is the collision of the transmissions from a subset of the K nodes. The nodes continue generating a random bit and transmitting it until the reader recovers the vector $\mathbf{x}$. At this point the reader triggers the nodes to stop transmitting which it can do by terminating its RF signal [16].

Corresponding Code: Let M be the length of the binary string that each node has transmitted before the reader terminates the process. Let $\mathbf{A}$ be the $M \times N$ random binary matrix where each column of $\mathbf{A}$ corresponds to the string transmitted by a particular node and each row of $\mathbf{A}$ corresponds to a time slot. Thus, $\mathbf{A}_{j,i} = 1$ if backscatter node i transmits in time slot j. The reader receives a vector $\mathbf{y}$ of M symbols where each symbol is obtained in a time slot. The vector $\mathbf{y}$ can be written as:

$$\mathbf{y} = \mathbf{A}_{M\times N}\mathbf{x}_{N\times 1} \tag{2}$$

Since $\mathbf{x}$ is sparse, i.e., it has only $K \ll N$ non-zero entries, compressive sensing theory tells us that we can efficiently estimate $\mathbf{x}$ with high accuracy given only a few symbols, specifically $M \approx K\log(N/K)$ [13, 9].

The above argument ignores the channels of the backscatter nodes. However, recall that backscatter nodes transmit in a narrow band ($\leq$ 640 KHz) [14] and hence their channels can be modeled as a single complex number (i.e., a single tap channel) [46]. Incorporating the channels, Eq. 2 can be rewritten as:

$$\mathbf{y} = \mathbf{A}\begin{bmatrix} h_1 x_1 \\ \vdots \\ h_N x_N \end{bmatrix} = \mathbf{A}\begin{bmatrix} h_1 & & \\ & \ddots & \\ & & h_N \end{bmatrix}\mathbf{x} = \mathbf{AHx} = \mathbf{Az} \tag{3}$$

where $\mathbf{H}_{N\times N}$ is a diagonal channel matrix and $\mathbf{H}_{i,i} = h_i$ is the complex channel coefficient of the i^{th} node. Since the vector $\mathbf{z} = \mathbf{Hx}$ is sparse and has only K non-zero entries, this is again a compressive sensing problem. In this case, the compressive sensing algorithm will estimate the sparse vector $\mathbf{z} = \mathbf{Hx}$, where $z_i = h_i$ if node i has data and 0 otherwise. This allows us to identify the nodes with data to transmit and estimate their channel coefficients.

5.1 Optimizing Performance

While the formulation of the core problem in Eq. 3 is efficient in terms of the number of time slots, the overhead on the reader is

[1] Generating random numbers is a routine operation carried out in today's RFID systems for node identification as required by the EPC Gen-2 standard [14], and does not introduce extra overhead on the nodes [3].

likely to be prohibitive. Specifically, to estimate $\mathbf{z} = \mathbf{Hx}$, compressive sensing requires the matrix $\mathbf{A}$ to be known at the decoder. In theory, the reader can generate this matrix by using the same pseudorandom number generator used by the nodes and feeding it with all ids in the network. However, in practice, this is computationally infeasible since the number of columns in $\mathbf{A}$ is equal to the number of nodes in the network, which can be huge. Also, even if the reader generates the matrix offline and stores it, it still cannot operate on it because of the computational complexity and the associated increase in the running time. Note that this is not an issue for the backscatter nodes because each node only needs to know the column of $\mathbf{A}$ corresponding to its own id, which it can generate using its id as a seed. Nodes that do not have data do not need to generate their columns in the matrix because the model multiplies those columns by zero. The reader however does not know a priori which nodes have data. It will obtain this information only after solving the matrix equation.

Thus, we transform the above compressive sensing problem into an equivalent compressive sensing problem, but with significantly reduced computational complexity. In doing so, we aim to make both the number of transmitted symbols and the computational complexity functions of K, the number of nodes with data, and independent of N, the total number of nodes in the network.

We start by focusing on a temporary id space. Each of the K nodes that want to transmit randomly picks a temporary id and uses it as its identifier for subsequent communication in the data transmission phase. The temporary id space can be much smaller than N since it is only used to distinguish between the K nodes that want to transmit. However, without knowing K we might define a space too small in which case some of the K nodes will pick the same temporary id and become indistinguishable, or an unnecessarily large space where the computational complexity is still high. Therefore, we first estimate K to properly set the size of the temporary id space. Then, we further reduce the scale of the problem by quickly eliminating large chunks of this temporary id space. Finally, we use standard compressive sensing to solve a small sparse recovery problem. In the rare event where the estimate of K causes an error, the reader starts over as is the case in today's RFID systems.

A. First Stage: Estimating K.

Inspired by prior work that uses streaming algorithms to efficiently count the number of active flows on high speed links [15], here we use a simple streaming technique to quickly estimate the number of backscatter nodes with data to transmit. Again we divide time into short slots that correspond to the transmission of one bit, i.e., one wireless symbol. Every s time slots constitute a single step. In the first step, and for each of the s time slots, each node with data transmits a "1" bit with probability 0.5 and stays silent with probability 0.5; in the second step and for each of the s time slots, each node will transmit "1" with probability 0.25; and so on. In step j and for each of the s time slots in that step, each node transmits a "1" bit with probability $p_j = 2^{-j}$.

The reader only distinguishes between two states of the time slots, "occupied" and "empty". This can be done based on the power level. The expected number of empty slots in each step is $s \times (1-p_j)^K$. The percentage of empty slots is $E_j = (1-p_j)^K$. Thus one can calculate an estimate of K as:[2]

$$\widehat{K} = \frac{\log E_{j^*}}{\log(1-p_{j^*})} \tag{4}$$

where j^* is the step that E_j exceeds a predefined threshold and the algorithm terminates. Once we reach a step in which the probability p_j is very low, the slots will become mostly empty. By setting the threshold for the percentage of empty slots close to 1, we can get a fairly tight estimate of K in $\log K$ steps. One can prove the following:

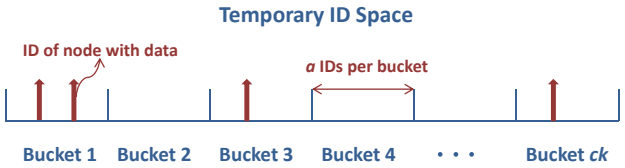


Figure 4—The temporary id space of size $a \times c \times K$ is hashed into cK buckets, each containing a ids. At most K buckets will have ids of the nodes of interest. Temporary ids hashed to empty buckets can be ruled out (e.g., ids hashed to buckets 2 and 4).

LEMMA 5.1. *There exists an absolute constant $C > 1$ such that for any $\epsilon, \delta \in (0,1)$, if the number of slots per step s is equal to $C\log(1/\delta)/\epsilon^2$, then with probability at least $1 - O(\log k \cdot \delta)$, we have $\widehat{K} = (1 \pm \epsilon)K$ and $j^* = \log K + O(1)$.*

We note the proof is similar in nature to the standard analysis of streaming and estimation algorithms as seen in [15, 32, 29]. Finally, we note that Buzz does not need an exact estimate of K. An approximate estimate with the guarantees in the above lemma is sufficient for achieving the performance of our identification protocol.[3]

B. Second Stage: Reducing the Scale of Compressive Sensing.

Once we have an estimate of K, we can reduce the scale of the problem from the size of the entire node population N to the size of a temporary id space, which is a function of K. Specifically, we set the size of this temporary id space to $a \times c \times K$. The parameters a and c will be used in the following sections to balance the tradeoff between transmission time and decoding complexity.

Only K of the $a \times c \times K$ temporary ids correspond to nodes that want to transmit. To further reduce the scale of this sparse recovery problem, we hash the $a \times c \times K$ possible ids into cK buckets. Each bucket will contain a ids as shown in Fig. 4. We represent these cK buckets by allocating cK time slots; each slot's duration is of the length of one bit. A backscatter node with data to transmit will transmit "1" in time slot j if its temporary id hashes to bucket j. The reader then checks the power in each time slot. We denote a bucket as empty if no power is detected in its corresponding time slot. Temporary ids that hash to empty buckets can be eliminated. Given that there are only K transmitting nodes, at most K buckets will be non-empty. Since each bucket contains a ids, at the end of this stage, at most aK ids remain as candidate node ids.

C. Third Stage: Compressive Sensing Decoding.

Now that we have reduced the scale of the compressive sensing problem to recovering K temporary ids out of only aK possible ids, we can apply the protocol described at the beginning of §5. Let $\mathbf{A}'$ be a reduced version of the matrix $\mathbf{A}$ that keeps only the columns corresponding to the remaining aK possible temporary ids; and $\mathbf{x}'$ and $\mathbf{H}'$ are the similarly reduced forms of $\mathbf{x}$ and $\mathbf{H}$. The reader only needs to regenerate $\mathbf{A}'$, as opposed to $\mathbf{A}$, and solve the system:

$$\begin{aligned} \mathbf{y} &= \mathbf{A}'\mathbf{H}'\mathbf{x} \\ &= \mathbf{A}'\mathbf{z}' \end{aligned} \tag{5}$$

[2]The numerator of Eq. 4 can be written as $\log \min(E_{j^*}, 1 - 1/s)$ to handle the case where all slots are empty i.e. $E_{j^*} = 1$.

[3]Having an estimate of the number of nodes is also beneficial to existing protocols. In §10, we feed our estimate of K to the protocol specified in the EPC Gen-2 standard, and show that Buzz still significantly outperforms it.

To decode, the reader uses compressive sensing to estimate the elements of vector $\mathbf{z}'$. In compressive sensing, estimating $\mathbf{z}'$ is formulated as an optimization problem [13, 9]:

$$\begin{aligned} \underset{z'}{\text{minimize}} \quad & \|\mathbf{z}'\|_1 \\ \text{subject to} \quad & \mathbf{A}'\mathbf{z}' = \mathbf{y}, \end{aligned} \tag{6}$$

where $\|.\|_1$ is the L1 norm.

Since the space of the problem is now aK as opposed to N, compressive sensing tells us that for a sparsity level $a = \omega(1)$ we only need to receive about $K\log(aK/K) = K\log(a)$ symbols to decode $\mathbf{z}'$ if elements in $\mathbf{A}'$ are randomly generated binary numbers [13].

D. Communication Cost and Decoding Complexity.

In terms of the communication cost, the first and second stages of Buzz's identification algorithm require $s \times j^* = s\log(K)$ and cK time slots (i.e., wireless symbols) respectively. The compressive sensing stage requires $M \approx K\log a$ time slots. Thus, the communication complexity of Buzz is $O(s\log(K) + cK + K\log a)$ bits.

In terms of decoding complexity, the running time of the first two stages is $s\log(K) + cK$. The third stage involves solving a compressive sensing problem for an $M \times aK$ matrix $\mathbf{A}'$, where $M \approx K\log a$. There is a large body of algorithms in compressive sensing that use linear programming to efficiently solve the problem in Eq. 6 [11, 10]. In our implementation, we use an algorithm based on the interior-point method [22][4], which has the running time of $T = O(I \cdot T_{A'})$, where $I \approx \sqrt{aK}$ is the number of iterations in the algorithm, and $T_{A'}$ is the cost of a matrix-vector multiplication involving $\mathbf{A}'$ [4]. For efficiency, we use sparse binary matrices $\mathbf{A}'$ for which $T_{A'} = O(aK\log(aK/K)) = O(aK\log a)$ [4]. This yields

$$T = O(I \cdot T_{A'}) = O(\sqrt{aK} \cdot aK\log a) = O((aK)^{3/2}\log a).$$

Therefore, the total decoding complexity of the three stages is equal to $O(s\log(K) + cK + (aK)^{3/2}\log a)$.

In our implementation of the K estimation stage, we set $s = 4$ and the termination threshold to 0.75, and use the estimated $\widehat{K}$ to set parameters for the latter stages. The latter two stages of the customized compressive sensing approach provide a trade-off between the decoding complexity and the communication cost. One should choose the parameters a and c properly based on system capabilities, constraints and requirements. In our experiments, we set c to 10 and $a = K$.

6. DISTRIBUTED RATE ADAPTATION

We describe a protocol that enables backscatter networks to adapt the uplink bit rate to the channel quality. In contrast to traditional work on rate adaptation [47, 25], which adapts the rate of individual senders, this protocol adapts the aggregate rate of the backscatter nodes that want to transmit. Said differently, we model the system as a single virtual sender that is realized by the collective transmissions of the backscatter nodes. We then design a protocol that optimizes the bit rate of the virtual sender given the channel conditions. The protocol operates in a distributed fashion and the reader does not schedule the transmissions/collisions of nodes.

(a) Protocol: In Buzz's rate adaptation protocol, the reader initiates the data transmission phase by sending a single command to all backscatter nodes. Upon decoding this command, each node starts a random binary number generator which is seeded by its own temporary id and the current time slot[5] and returns a random bit. In each time slot, if the random bit generated is "1" the node will transmit its message, otherwise the node remains silent. Hence, depending on the results of the random binary number generator, random subsets of the nodes will collide together. The nodes repeat this process (generating a random bit and transmitting the message if the bit is "1") until the reader signals them all to stop.

The reader receives the colliding messages and tries to decode them (as described below). Once it has decoded all K messages and the decoded messages pass the CRC check, the reader triggers the backscatter nodes to stop transmitting the current messages (and if applicable move on to the new message). The reader can do this by terminating its RF signal, which naturally terminates all backscatter transmissions [16]. Note that the reader does not send signals to indicate the beginning or end of a time slot. The nodes automatically move on to the next time slot until there is no RF power to reflect.

(b) Encoder: The result of the above protocol is a rateless code. For clarity, we first express the code assuming that each of the K messages is one bit. We use the vector $\mathbf{b}$ to denote the K bits that the K nodes transmit, where $b_i \in (0,1)$ is the bit of node i. When transmitted according to the above protocol, these bits will collide randomly on the channel and the reader will receive the vector $\mathbf{y}$:

$$\mathbf{y}_{L\times 1} = \mathbf{D}_{L\times K}\mathbf{H}_{K\times K}\mathbf{b}_{K\times 1} \tag{7}$$

where $\mathbf{H}$ is a diagonal matrix whose diagonal element $h_{i,i}$ represents the channel of backscatter node i; $\mathbf{D}$ is a binary matrix, where element $d_{j,i}$ is the j^{th} random number used by node i in the above protocol; L is the number of time slots until the reader decodes all K bits and triggers the nodes to stop. Note that while $\mathbf{b}$ is a *binary* vector that corresponds to the K bits transmitted by the backscatter nodes, $\mathbf{y}$ is a vector of complex numbers that refer to the L wireless symbols received at the reader.

We generalize the above to P-bit messages as follows:

$$\mathbf{Y}_{L\times P} = \mathbf{D}_{L\times K}\mathbf{H}_{K\times K}\mathbf{B}_{K\times P}. \tag{8}$$

The only difference from Eq. 7 is that now both the received and the transmitted signals are expressed as matrices.

(c) Belief Propagation Decoder: The reader knows $\mathbf{H}$ from the node identification step. It can generate $\mathbf{D}$ because it also knows the ids of the nodes, i.e., seeds, and has the same random number generator they use. It knows $\mathbf{Y}$ because this is the collision signal that it receives. The task is to retrieve the original messages $\mathbf{B}$.

One approach to decode is inverting the encoding matrix $\mathbf{M}_{L\times K} = \mathbf{D} \times \mathbf{H}$ (if $L \neq K$ one can use the pseudo inverse) [46]. This solution is undesirable because it does not take into account the reader's knowledge that the matrix $\mathbf{B}$ is binary. It is also highly expensive from a computational standpoint. To provide the rateless property, the reader has to invert the matrix for each value $L' = 1, \ldots, L$ until it finds the right L' at which all messages pass the CRC check. Thus, instead of inverting the matrix, Buzz uses a belief propagation algorithm that decodes incrementally and also embeds the knowledge that the messages are binary strings.

We will explain how Buzz decodes the j^{th} bit transmitted by all nodes. The same procedure follows for decoding all the remaining bits in the message. The j^{th} bit of each node collides only with the j^{th} bits of other nodes and can be decoded separately from the other bits in the message. Hence, we will decode using Eq. 7.

Eq. 7 can be modeled as a bipartite graph shown in Fig. 5 where the K vertices on the left represent the original bits $\mathbf{b}$, and the vertices on the right represent the received symbols $\mathbf{y}$. There is an edge from b_i to y_j if $d_{j,i} = 1$, the weight of which is $h_{i,i}$. Hence, the set

[4] Faster algorithms exist in the literature to solve the standard compressive sensing problem such as [5].

[5] Unlike previous stages, in the uplink data transmission case, the duration of each time slot is the length of the nodes' messages. For simplicity we assume it is fixed.

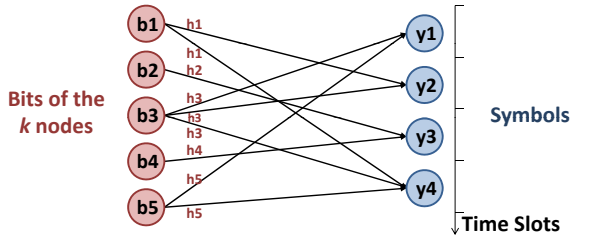


Figure 5—Belief Propagation Bipartite Graph: On the left are the bits transmitted by the backscatter nodes. On the right are the received symbols. An edge represents a node's contribution to a received symbol, i.e., the node participates in the collision on that symbol. If b_i is flipped in the belief propagation decoding process, only the gains of bits that have collided with i will be updated.

1 Pseudocode for jointly decoding one bit of all nodes

INPUT:
$\mathbf{y}, \mathbf{H}, \mathbf{D}$
OUTPUT:
Decoded bits $\hat{\mathbf{b}}$
PROCEDURE:
Initialization:
$\hat{\mathbf{b}} \leftarrow K$ independent random bits
Belief propagation decoding:
Calculate gains $G_1, G_2, \ldots, G_K$ using Eq. 9
repeat
  Find i s.t. $G_i = \max G_1, G_2, \ldots, G_K$
  $\hat{b}_i \leftarrow \hat{b}_i \bigoplus 1$
  Update G_i and the gains of node i's neighbors' neighbors
  if $G_1, G_2, \ldots, G_k \leq 0$ **then**
    Done
until Done

of incoming edges to y_j identifies all bits whose collision creates y_j weighted by their channels. The set of outgoing edges from a bit b_i identifies all collision symbols to which it has contributed.

The objective of the belief propagation algorithm is to find a vector $\hat{\mathbf{b}}$ that could have produced the received signal $\mathbf{y}$, or more concretely to find $\hat{\mathbf{b}}$ that minimizes the error:

$$\min_{\hat{\mathbf{b}}} \|\mathbf{DH}\hat{\mathbf{b}} - \mathbf{y}\|_2.$$

The algorithm works as follows. Initialize $\hat{\mathbf{b}}$ to a random binary vector. In each iteration, flip one bit in $\hat{\mathbf{b}}$ that best minimizes the remaining error $\|\mathbf{DH}\hat{\mathbf{b}} - \mathbf{y}\|_2$ and keep flipping until the error cannot be reduced any further by flipping any single bit, i.e., reaching an optimum.

To quickly find the bit, which once flipped would yield the biggest gain, Buzz maintains for each bit i in $\hat{\mathbf{b}}$ a variable G_i, representing the gain achieved by flipping bit i, i.e., the reduction in error. G_i is calculated as follows:

$$\begin{aligned} \tilde{\mathbf{b}} &= \hat{\mathbf{b}} \\ \tilde{b_i} &= \hat{b_i} \oplus 1 \\ G_i &= \|\mathbf{DH}\hat{\mathbf{b}} - \mathbf{y}\|_2 - \|\mathbf{DH}\tilde{\mathbf{b}} - \mathbf{y}\|_2 \end{aligned} \tag{9}$$

At the beginning of the first iteration, G_i is calculated based on the randomly initialized $\hat{\mathbf{b}}$. Then after flipping bit i of $\hat{\mathbf{b}}$, Buzz updates G_i and the gains of all nodes which have collided with node i, i.e., update G_l if the i^{th} column and l^{th} column of $\mathbf{D}$ have 1 in at least one same row. This could be more clearly seen in the bipartite graph. After flipping bit i of $\hat{\mathbf{b}}$, only the gains of the bits that are neighbors of neighbors of the vertex $\hat{\mathbf{b}}_i$ in the bipartite graph need to be updated, as other bits are not affected. For example, in Fig. 5 if bit 4 was flipped, we only update the gains of bits 2 and 4. If bit 1 was flipped, we update the gains of bits 1, 3 and 5. A pseudocode for decoding one bit is shown in Alg. 1.

The algorithm finishes its decoding of the j^{th} bits in the messages once all gains are negative. It then moves on to the bits at the $j+1$ position. After decoding all positions, the algorithm checks whether the resulting messages pass the CRC check. If any message passes the CRC check, the values of the bits in the message will be fixed for all subsequent decoding. The reader keeps collecting collisions until all decoded messages pass the CRC check.

(d) Discussion: A few points are worth elaborating on:

- We choose the matrix $\mathbf{D}$ to be sparse, i.e., each row of $\mathbf{D}$ has only a few ones but is mostly zero. A sparse $\mathbf{D}$ reduces the decoding time because only a few gains need to be updated when a bit is flipped. It also allows the reader to decode the messages from fewer collisions because it reduces the number of local minima of the error function. This argument is similar to why belief propagation works well with low density parity check code [20]. Hence, Buzz uses a low density code by making the matrix D sparse. The sparsity of D is related to K, the number of nodes that want to transmit. The reader communicates its estimate of K to the nodes, which they give as an input to the random binary number generator to bias its output.
- Due to the near-far effect, different nodes may have very different channel coefficients. Unlike in CDMA, the near-far effect is mitigated in Buzz because: first, the code is sparse (only a few nodes collide at a time); second, each time, a node's signal combines with a different subset of nodes, providing diversity. The results in §9 and §10 are in the presence of near-far effect. Further, this property can be leveraged to improve decoding and reduce the overall time. Specifically, bits of nodes with good channels will have the largest gains and will directly converge to the right binary values in the first few iterations. Once they are decoded and pass the CRC check, we set their gains to be negative infinite so that flipping the bits of nodes with worse channels in later iterations will not flip them back to a wrong value. This effectively cancels their contributions to the collisions.
- If a backscatter node runs out of power in the middle of the data collection phase, its impact on the other nodes will be minimal. Specifically, the nodes which have already been decoded will not be affected. This node's influence on the system translates to additional noise to the collisions it participated in, i.e., it will take more collisions to decode the remaining nodes.
- Buzz achieves its goal of delivering a rateless code that adapts to channel conditions. Specifically, if the decoding process finishes in L time slots with zero bit error, it means $K \times P$ data bits are successfully delivered using $L \times P$ symbols. Hence, the rate of the system is $\frac{K}{L}$ bits/symbol. When the channels are good, then we have $L < K$ and the system delivers multiple bits per symbol. In contrast, in current backscatter systems, nodes transmit sequentially, thus making the highest rate 1 bit/symbol regardless of the quality of the channel [14]. Furthermore, in current systems if the channel of some node is bad and cannot support 1 bit/symbol, decoding will fail. In contrast, with Buzz, a node with a bad channel will just take more collisions to decode. This rateless design eliminates the need for frequent feedback as required in today's RFID systems, reducing the overhead on both the nodes and the reader.

7. IMPLEMENTATION

We build a prototype of Buzz using USRP software radio [28] as the reader and the UMass Moo computational RFIDs [50] as backscatter nodes in the network.

Reader: We adopt a USRP implementation of an EPC Gen-2 RFID reader developed in [8] and customize it to incorporate Buzz's node

(a) UMASS Moo Computational RFID

(b) General Purpose Commercial RFID

Figure 6—We experiment with two types of RFID tags: (a) The Moo computational RFIDs with programmable microcontrollers and (b) the Alien Squiggle General Purpose UHF RFID Tags.

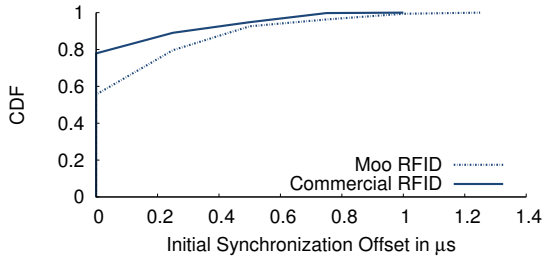


Figure 7—CDF of the initial synchronization offset between different RFID tags: Both commercial and computational RFIDs have an initial offset of less than 1μsec which has negligible impact on the performance of Buzz.

identification and distributed rate adaptation algorithms described in §5 and §6. This mainly includes changing the reader commands to allow backscatter nodes to distinguish which algorithm to run. Due to the overhead of user mode processing in software radios, we do not run the decoders in real time. Instead, we collect the traces from the USRP receiver and process them offline.

UHF backscatter systems operate between 860 MHz and 960 MHz [14]. We use RFX900 USRP daughterboards and Cushcraft 900 MHz RFID antennas which operate in this range [34]. We run all experiments at a carrier frequency of 925 MHz. The USRP reader transmits its queries at a data rate of 27 kbps and captures measurements of the tags' signals at 4 MHz.

Backscatter Nodes: The Moo tag, shown in Fig 6(a) is a passive computational RFID that operates in the UHF band [50]. It has an onboard temperature sensor, accelerometer, and a programmable MSP430F2618 microcontroller [50]. The original code base implemented in the Moo already supports most parts of the EPC Gen-2 standard and has all components necessary for Buzz, including decoding the reader's commands, reading data from memory and transmitting it in a random time slot, and changing the impedance of the antenna to transmit zeros and ones [50]. Since Buzz's algorithms require randomization in many stages, we randomize the encoding process offline and program 20 different randomly encoded traces into each Moo tag in advance. This allows the tags to transmit different data in each run without being reprogrammed.

In addition to the Moo, we also experiment with commercial RFIDs (The Alien Squiggle Inlay [1]) shown in Fig. 6(b). We use them to study the level of synchronization in general purpose RFIDs as we will discuss in §8.1.

Setup: We deploy the tags on a movable plastic cart placed on a 1.5m×3m table, where the USRP reader also sits. Distances between tags and reader vary in a range of [0.5,6] feet. The maximum distance is limited by the capabilities of the Moo, whose typical operating range is 2 feet [50].

8. MICROBENCHMARK

We start with a few experiments that provide insight into the working of the system.

8.1 Synchronization

Buzz assumes that transmissions of backscatter nodes are synchronized. In this section, we present measurement findings for both commercial RFIDs and computational RFIDs, and show that the synchronization accuracy in practice is sufficient for Buzz's performance. In particular, synchronization in backscatter networks has two aspects:

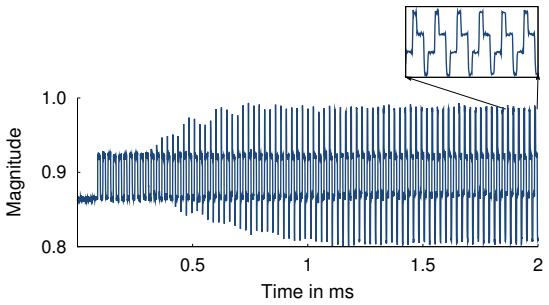
- *The starting offset of the nodes' transmissions:* Backscatter nodes are naturally synchronized because they are triggered by the reader's signal. However, the jitter in detecting the reader's signal can lead to initial offsets.
- *Drift in the nodes' digital clocks:* Backscatter nodes use a digital clock to time their operations. Different clocks have different drifts, which can be corrected by having each node estimate its clock drift relative to a virtual clock maintained by the reader and compensate for the difference. This is done by counting the number of clock ticks between two pulses from the reader separated by a fixed time interval.

Initial Offset: We measure the offset using two types of off-the-shelf passive UHF RFID tags: Alien Squiggle commercial RFIDs [1] and the Moo computational RFIDs [50]. We let the tags concurrently reply to a reader's query and measure the offset between the starting points of their transmissions. We use 20 different tags of each type and vary the number of concurrent tags in each run between 2 and 8.

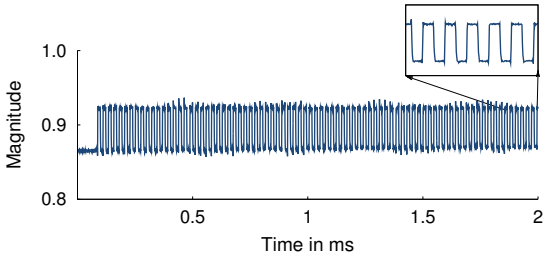
Fig. 7 shows the CDF for the measured offsets. The offset's 90^{th} percentile is 0.3 μs for the commercial tags and 0.5 μs for the Moo, and the maximum is less than 1 μs. The default bit rate used in practical settings and recommended by manufacturers is 64 kbps [2, 27]. At such rates, a 1 μs offset is about 6.5% of the bit length (16 μs). We show in §9 that its impact on performance is negligible.

The EPC Gen-2 standard allows for a maximum bit rate of 640 kbps [14]. Although at such maximum rate the offset can be 65% of the bit length, the reader can sample the backscatter signal at a much higher rate and obtain many samples per bit. The reader can then use the middle samples of each bit to increase robustness to synchronization errors, which affect the samples at the boundary of a bit. For instance, with a sampling rate of 20 MHz (which is simple with today's hardware capability [28]), the reader can get 10 synchronized samples of each bit when the tags transmit at the maximum bit rate of 640 kbps. Alternatively, one can also design the circuit of the backscatter nodes to further improve synchronization accuracy [12, 23, 41].

Drift Correction: To test how well backscatter nodes can correct for their clock drift, we program the Moo tags to use a reader's virtual clock and insert extra clock cycles to correct for the drift. We let two tags transmit the same data stream at 80 kbps and capture the trace on the USRP reader with and without correcting for the clock drift. Fig. 8 shows a trace for which the initial synchronization offset is zero to help visualize the effect of clock drift over time. After 2 ms ($\approx$ 160 bits) in Fig. 8(a), the clock drift causes the bits of the two tags to be misaligned by 50% of the symbol length. Fig. 8(b) shows that after correcting for the clock drift, the bits of the two tags are still aligned after 2 ms. The results reported in this paper use this method for drift correction. We note that the clock drift of each tag is fairly stable [12, 50]. We let each Moo tag compute its clock drift



(a) Collision Without Correction for the Clock Drift



(b) Collision after Correcting for the Clock Drift

Figure 8—Two tags concurrently transmit the same data: (a) Without correcting for the clock drift, the bits of the two tags are misaligned by 50% of the symbol length after 2 ms. (b) When the tags correct for their clock drifts, their bits remain aligned.

once and used the same estimate throughout our experiments over a period of four months.

8.2 Understanding the Belief Propagation Decoder

To better understand how Buzz's belief propagation algorithm decodes collisions to achieve distributed rate adaptation, let us zoom in on one example where 14 Moo tags had messages to send and the messages were decoded in ten slots. In line with the EPC Gen-2 standard, each tag transmitted a 96-bit message at a bit rate of 80 kbps [14].

Fig. 9 shows how the average bit rate adapted over time. For each time slot, the figure shows the number of tag messages already decoded in previous slots as a dark blue bar and the number of newly decoded tag messages upon the arrival of the current time slot as a light blue bar. The decoding process started off strong and quickly reached its peak. Eleven tag messages were decoded within the first four slots, resulting in a peak bit rate of 2.75 bits/symbol. The decoding of these eleven tags can generally be categorized into two types. Certain tags had a very good channel and were therefore immediately decoded from the first collision they participated in, such as the two tags decoded in the very first time slot. In contrast, other tags' messages were uncovered as the belief on their data gradually built up when the tags they collided with were decoded. This process resembled a ripple effect, resulting in the high decoding efficiency in slot 3 and 4.

After slot 4, it took Buzz another six slots to decode the three remaining tags (in the same experiment, TDMA and CDMA consistently failed to decode the last two tags' messages). Notably, the last tag participated in four collisions to finally get decoded. Because of this particular tag, the average bit rate was dragged down and other tags had to transmit an extra number of times. By the end of the transfer, the aggregate bit rate was 1.4 bits/symbol.

This experiment also shows a limitation of Buzz. Namely, some nodes may transmit after they have been decoded. This is due to the fact that Buzz treats all nodes with data to transmit as a single distributed sender. An alternative design choice could have the

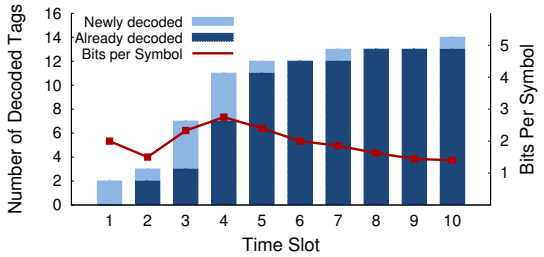


Figure 9—The progress of Buzz's decoding of 14 tags during 10 time slots: Buzz quickly decoded 11 tags within the first 4 slots, and then gradually adapted the aggregate bit rate to channel conditions to decode all remaining tags.

reader ACK the tags whose messages have been decoded, so that they remain silent in subsequent time slots. To silence the tags, the reader would at least need to ACK by echoing the temporary id of each decoded tag. A back-of-the-envelope calculation based on the EPC Gen-2 standard suggests that a 75% overhead on top of the uplink transmission time is necessary to silence these 14 tags [14]. Therefore, the cost of silencing individual tags seems to outweigh the benefits, and so Buzz avoids such a design.

9. Results for Uplink Transmission

In this section, we focus on the efficiency and reliability of the uplink transmission phase in backscatter networks. We assume that the backscatter reader has already performed node identification and resource allocation (evaluated in §10). We compare Buzz with the following two baseline schemes:

- **TDMA:** Tags sequentially transmit one after another. In line with the EPC Gen-2 standard, each tag uses a bit rate of 80 kbps for both TDMA and Buzz. Miller-4 code is used in TDMA to increase its robustness [14].
- **CDMA:** We compare Buzz with synchronous CDMA using Walsh codes, because asynchronous CDMA severely suffers from the near-far problem [30].[6] Walsh codes are the most commonly used orthogonal codes in CDMA systems. For K tags, we use a set of Walsh codes with a spreading factor of K. We let CDMA use the same symbol rate as Buzz, 80k symbols per second, and adopt a standard CDMA decoder for orthogonal codes [30] to decode the received CDMA signal.

We run the experiments for $K = 4$, 8, 12, and 16 Moo tags. For each run of all experiments, the tags decode which scheme to use from the reader signal (TDMA, CDMA, or Buzz), and transmit a 32-bit message with a 5-bit cyclic-redundancy check. We repeat the experiments at ten different locations for each value of K. At each location, we run the three schemes to collect five traces each, immediately one after another without changing the environment, i.e., moving the tags or reader antennas.

Transmission Efficiency: Fig. 10 shows the time it takes the three schemes to complete data transfer. TDMA and CDMA have a fixed aggregate bit rate, and therefore always finish transmission in a fixed amount of time for each value of K. In contrast, Buzz only completes the data transmission phase when the reader has successfully decoded the messages of all tags. Hence the total transfer time for Buzz is variable. As we can see, Buzz finishes in approximately half the time of TDMA and CDMA on average,[7] translating

[6] In cellular CDMA systems, the near-far problem is addressed by adjusting the power level at the transmitters [30], which is not feasible for backscatter nodes.

[7] The bump on the CDMA curve at $K = 12$ is due to the lack of

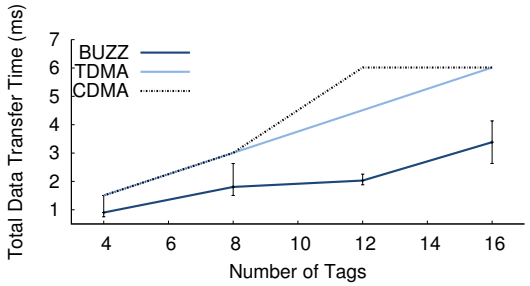


Figure 10—Total data transfer time: Buzz achieves 2× improvement in total transfer time over TDMA and CDMA. Unlike TDMA and CDMA which have a fixed rate of 1 bit/symbol, the rateless nature of Buzz allows it to adapt to an average rate of 2 bits/symbol.

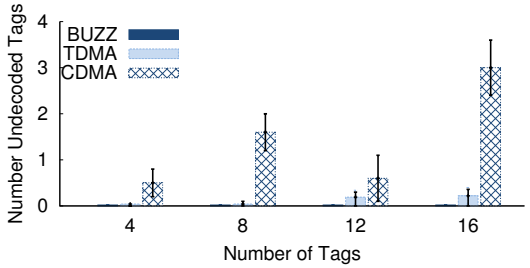


Figure 11—Message errors: CDMA has the lowest reliability. TDMA has few errors because it uses Miller-4 code. Buzz's bars are not visible because it has zero errors due to its rateless code.

to a 2× increase in aggregate bit rate of the network. However, this statistic understates the efficiency gain Buzz brings about. Unlike Buzz, TDMA and CDMA do not adapt their rates, and therefore might have decoding errors upon the completion of all the nodes' transmissions. These decoding errors lead to retransmissions, further reducing their efficiency.

Message Reliability: Commercial backscatter systems use a checksum to decide whether a message is decoded correctly and send a NAK if the checksum does not pass [14]. Thus, the number of messages decoded incorrectly represents a lower bound of retransmissions needed. Fig. 11 shows the number of tags whose messages were not correctly decoded for the same traces in Fig. 10.

Due to its automatic rate adaptation, Buzz decodes all messages correctly. TDMA has very few errors because it uses Miller-4 code which increases its robustness to bad channels. CDMA is less reliable, and as the number of tags in the network increases, its reliability quickly degrades. This is mainly because all tags get equal shares of the channel in CDMA, despite the fact that there is generally a disparity in channel quality between them. Note that in Fig. 11, CDMA for $K = 12$ has fewer undecoded messages than both $K = 8$ and $K = 16$. The reason for this is that we let each of the twelve tags use unique Walsh code of 16 bits, as no Walsh code of 12 bits is available [30]. This implies that one could use a longer code to increase reliability in CDMA, however, this comes at the expense of efficiency.

Next, we focus on more challenging channel conditions to further investigate the reliability of Buzz and TDMA. In this experiment, we use four tags and keep moving the tags further and further away from the reader to worsen the channels of all tags. Fig. 12 shows the comparison between Buzz and TDMA. When the channel quality in the network is good, Buzz decodes all four tags' messages correctly in a single time slot (4 bits/symbol), while TDMA requires 4 time slots to decode. As the channel worsens, TDMA starts failing to

a set of orthogonal Walsh codes of length 12 [30]. So instead, for $K = 12$ we use the orthogonal Walsh codes of length 16.

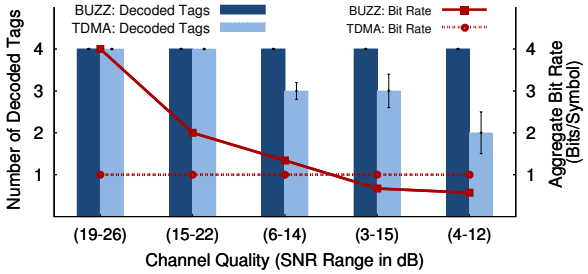


Figure 12—Under challenging conditions where TDMA experiences 50% (median) of message loss, Buzz adapts the bit rate to below 1 bit/symbol to match the channel quality and is able to decode with zero errors.

decode while Buzz gradually adapts the aggregate bit rate to below 1 bit/symbol. Notably, in the most challenging channel conditions, TDMA decodes two tags' messages incorrectly out of four, experiencing a 50% message loss rate. It is worth noting that for the same tag placement, CDMA has a message loss rate of 100%. In contrast, Buzz adapts the aggregate bit rate of the network to 0.57 bits/symbol, taking seven time slots to correctly decode all tags' messages. This demonstrates Buzz's ability to adapt to a bit rate of below 1 bit/symbol under challenging channel conditions, which is important for improving reliability of backscatter systems.

Power Efficiency: We investigate Buzz's impact on energy consumption of the backscatter nodes, and compare it to TDMA and CDMA. Backscatter energy consumption however is ultra-low, and therefore if we measure it after individual message transmission, the measurement noise may potentially exceed the consumed energy. Thus, to obtain robust measurements, we make each scheme reply to a large number of queries from the reader and measure the total energy consumption. The Moo has a small capacitor, leaving us unable to measure this accumulative energy consumption. As a workaround, we attach a large capacitor ($C = 0.1F$) to the Moo so that it can store enough energy in advance for our experiment.

In each experiment, the USRP reader sends a sequence of 8800 queries spaced by 10.2 ms to $K = 8$ tags.[8] Upon receiving each query, each tag wakes up and replies its message using the three schemes. We measure the energy consumed by the transfer as:

$$E_{consumed} = \frac{1}{2}CV_0^2 - \frac{1}{2}CV_f^2, \tag{10}$$

where V_0 is the voltage on the backscatter node's capacitor before the transfer and V_f is the voltage at the end of the transfer. We measure $E_{consumed}$ for three different values of V_0 since the amount of energy spent by a node depends on its initial voltage. For each value of V_0, we repeat the experiment for ten different positions of the tags and reader. At each position, we run the schemes one after another without changing the environment.

Fig. 13 shows the average net energy consumption for a single run (averaged by 8800) across tags and positions. As expected, each tag transmits for a much longer time in CDMA, and hence consumes more energy. In contrast, independent of the starting voltage V_0, Buzz does not consume much more energy than TDMA. There are two primary reasons for this. First, the rateless code in Buzz is sparse. Each node only transmits its message for very few times. Second, although in TDMA each node transmits its message

[8]Backscatter nodes continue to harvest energy from the reader's RF signal even while they drain energy performing the operations of the three schemes. To ensure a fair comparison, we keep the duration of the reader's continuous waveform the same between queries for all three schemes.

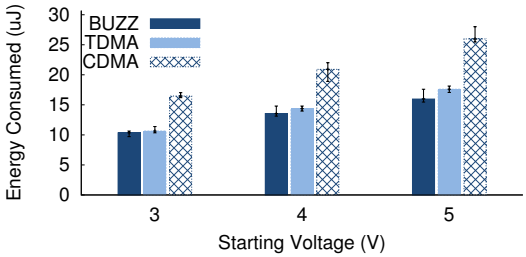


Figure 13—Energy consumption: Buzz does not consume much more energy than TDMA. Both Buzz and TDMA are significantly more energy efficient than CDMA.

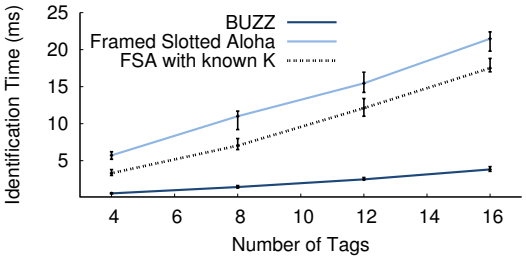


Figure 14—Identification time: Compared to Framed Slotted Aloha used in commercial RFID solutions, Buzz is significantly more efficient in identifying the nodes that want to transmit. Using the estimate of K from Stage 1 of Buzz's identification protocol improves the efficiency of Framed Slotted Aloha by 20%-40%.

only once, TDMA uses Miller-4 codes to protect the messages as recommended and hence has to switch the impedance on the antenna at 8 times of the data rate, which consumes more energy. If TDMA did not use Miller-4, it would lose its robustness and experience higher error rates [7]. Consequently, Buzz does not incur additional energy overhead to improve the efficiency and reliability of backscatter systems.

10. IDENTIFYING THE BACKSCATTER NODES

Recall that Buzz's node identification protocol consists of three stages: 1) estimating the number of backscatter nodes with data, 2) reducing the scale of the compressive sensing problem, and 3) compressive sensing decoding. In this section, we compare the efficiency of Buzz's three-step protocol with two baselines.

- **Framed Slotted Aloha (FSA):** This is the scheme adopted in the EPC Gen-2 standard [14]. Here, the reader initiates the identification phase by sending a query and allocating 2^Q time slots. Each tag picks a random slot and transmits a 16-bit random number (RN16), which serves as its temporary id. If a tag's id is correctly decoded (i.e., no collision), the reader will ACK this 16-bit temporary id. The reader adjusts Q to better accommodate the remaining active tag population, and repeats the procedure until all tags are identified. We follow the Q adjustment algorithm described in the standard [14], which initially sets $Q = 4$. It increases Q to $Q + C$ when a collision is detected, and reduces Q to $Q - C$ when no tag replies. We use $C = 0.3$ as recommended by the standard.
- **FSA Augmented with Estimated K:** We feed $\widehat{K}$, the estimate of K derived from Stage 1 of Buzz's identification protocol to FSA. Once FSA knows $\widehat{K}$, it sets $Q = log_2\widehat{K}$, i.e., allocating $\widehat{K}$ slots, instead of $Q = 4$ as the initial value. The maximum throughput for FSA is known to be $\frac{1}{e} = 36.8\%$, and is achieved when the number of allocated slots is equal to the number of nodes [8, 16]. Hence, setting $Q = log_2\widehat{K}$ improves the efficiency of FSA. Further, knowing $\widehat{K}$ can help FSA work with a smaller temporary id space as in Buzz. Specifically, we let each tag in FSA transmit a shorter random temporary id in the slot it picks instead of the RN16. This reduces both the uplink and downlink transmission time in FSA.

We run the experiment for $K = 4$, 8, 12 and 16 Moo tags, and repeat for ten locations each. Fig. 14 shows the total amount of time spent on identifying the tags, including the overhead for ACKing the tags. While in Buzz, the reader only sends a single signal to tell all tags to stop, in the compared schemes, each tag needs to be ACKed individually.

For 16 tags, using Buzz's compressive sensing scheme achieves a 5.5× reduction in identification time over original FSA, and is 4.5× more efficient than FSA with estimated K. Also, as we can see, using the estimate of K from Stage 1 of Buzz reduces the identification time in FSA by 20%-40%. This is mainly because the size of the temporary id space is reduced to a function of K, instead of 2^{16}. In conclusion, by compressing the sparse identity space and reducing the reader feedback overhead, Buzz reaches the goal of significantly speeding up the node identification phase in backscatter systems. Combined with the 2× throughput gain in data transmission, Buzz reduces the overall communication time by 3.5×.

11. RELATED WORK

Various protocols have been proposed to improve the performance of backscatter communication [31]. Most of them are based on TDMA, such as Framed Slotted Aloha adopted by the EPC Gen-2 RFID protocol [14] and the binary search tree algorithm [31]. Researchers have also studied the use of CDMA in this scenario [37]. FDMA and SDMA based schemes are proposed in [36, 49]. The common problems with these anti-collision protocols are twofold. First, since collisions (over time, frequency, code or space) can never be efficiently eliminated, these systems still end up wasting resources over collisions. Second, they divide the medium evenly among all backscatter nodes which typically have diverse channels.

In the context of ultra-low power systems (e.g., passive RFIDs), [43] is the only piece of work similar to ours in the sense that it also aims to decode collisions. However, it does not reduce identification overhead, or address the rate adaptation problem, but instead focuses on decoding repeated collisions in a low frequency proximity card environment. Further, the heavy machine learning decoders employed in [43] are not practical in high throughput systems.

There has recently been a lot of work on decoding collisions in WiFi networks to increase throughput [26, 21]. However, applying these techniques to backscatter networks is challenging. Successive interference cancellation requires exponential differences in the power levels of colliding nodes [26]. ZigZag decoding requires an offset between colliding messages [21], which is inefficient given the short messages in backscatter systems.

In the area of compressive sensing, the closest to our work is an algorithm in [40] which eliminates chunks of the space to improve running time. However, it uses complex deterministic codes which are not applicable to low power backscatter networks. [17] provides a theoretical analysis of using compressive sensing for sparse detection in on-off random access channels. It requires nodes to transmit arbitrary values chosen from a Gaussian distribution, which is infeasible since backscatter nodes can only transmit binary values.

Belief propagation in a sparse scenario is best-known for its use in LDPC codes [20]. Our belief propagation algorithm is similar to a bit flipping algorithm introduced in [33]. However, the algorithmic novelty of Buzz lies in distributedly coding the bits of mutiple transmitters, on the air. Accordingly, Buzz's bit flipping algorithm

is devised to effectively decode such a rateless code on a complex constellation graph representing multiple sources, unlike bit flipping decoding in the modulo-2 domain often employed by the coding theory community [33, 44].

Lastly, the rate adaptation problem in wireless networks has been extensively studied. Our design is inspired by recent advances in automatic rate adaptation using rateless codes, such as spinal and strider codes [39, 25], but differs in that we adapt the aggregate bit rate of the network through the design of a sparse distributed rateless code that is easy to decode using belief propagation.

12. CONCLUSION

For ultra-low power backscatter networks to reach the stage of widespread deployment, issues of reliability and efficiency must be addressed. This paper addresses these issues by modeling uplink transmissions in backscatter networks as if they were performed by a single virtual sender and treating collisions as a *sparse rateless* code across the nodes. We introduce a novel compressive sensing algorithm which significantly speeds up backscatter node identification and a belief propagation algorithm which enables distributed rate adaptation in data transfer. Evaluation of the new design shows a large improvement in both reliability and efficiency of backscatter communication.

Acknowledgments: We thank Nate Kushman, Ben Ransford, Shane Clark, Fadel Adib, Arthur Berger, Yu-Chih Tung, Phillip Nadeau, Shyamnath Gollakota, Hong Zhang, the reviewers and our shepherd, Michael Mitzenmacher for their insightful comments. We also thank Omid Aryan for helping with the experimental setup. This research is funded by NSF and the Interconnect Focus Center. We thank the members of the MIT Center for Wireless Networks and Mobile Computing, including Amazon.com, Cisco, Intel, Mediatek, Microsoft, and ST Microelectronics, for their interest and support.

13. REFERENCES

[1] Alien Technology Inc. ALN-9640 Squiggle Inlay. www.alientechnology.com.
[2] Alien Technology Inc. Common RFID Implementation Issues. Tech. Report. http://www.alientechnology.com/docs/.
[3] P. Bardell, W. McAnney, and J. Savir. *Built-In Test for VLSI: Pseudorandom Techniques*. John Wiley & Sons, 1987.
[4] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss. Combining geometry and combinatorics: a unified approach to sparse signal recovery. *Allerton Conference*, 2008.
[5] R. Berinde and P. Indyk. Sequential sparse matching pursuit. *Allerton Conference*, 2009.
[6] M. Brazeal. *RFID: Improving the Customer Experience*. Paramount Market Publishing, 2009.
[7] M. Buettner and D. Wetherall. A Gen 2. RFID Monitor Based on the USRP. *SIGCOMM Communication Review*, 2010.
[8] M. Buettner and D. Wetherall. A Software Radio-based UHF RFID Reader for PHY/MAC Experimentation. *IEEE RFID*, 2011.
[9] E. Candès, J. Romberg, and T. Tao. Stable signal recovery incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 2006.
[10] E. Candes and T. Tao. Near-optimal signal recovery from random projections and universal encoding strategies. *IEEE Transactions on Information Theory*, November 2004.
[11] I. Carron. Compressive sensing: Section 4 sparse recovery solvers. *http://sites.google.com/site/igorcarron2/cs*, 2012.
[12] N. Cho, S.-J. Song, S. Kim, S. Kim, and H.-J. Yoo. A 5.1-μw UHF RFID tag chip integrated with sensors for wireless environmental monitoring. In *ESSCIRC*, 2005.
[13] D. Donoho. Compressed sensing. *IEEE Trans. on Info. Theory*, 2006.
[14] EPCglobal Inc. EPCglobal Class 1 Generation 2 V. 1.2.0. http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.
[15] C. Estan, G. Varghese, and M. Fisk. Bitmap algorithms for counting active flows on high speed links. In *IMC*, 2003.
[16] K. Finkenzeller. *RFID Handbook*. John Wiley & Sons, 2010.
[17] A. Fletcher, V. Goyal, and S. Rangan. A sparsity detection framework for on-off random access channels. In *ISIT*, 2009.
[18] Frost & Sullivan. Global RFID healthcare and pharmaceutical market. Industry Report, 2011.
[19] Frost & Sullivan. Global RFID market. Industry Report, 2011.
[20] R. Gallager. Low-density parity-check codes. *IEEE Transactions on Information Theory*, 1962.
[21] S. Gollakota and D. Katabi. ZigZag decoding: Combating hidden terminals in wireless networks. In *SIGCOMM*, 2008.
[22] M. Grant, S. Boyd, and Y. Ye. CVX: Matlab software for disciplined convex programming. http://cvxr.com/cvx.
[23] R. E. Greeff, F. W. Smith, and D. K. Ovard. RFID device time synchronization. Patent US7889083, 2006.
[24] J. Griffin and G. Durgin. Complete link budgets for backscatter-radio and RFID systems. *IEEE Antennas and Propagation Magazine*, 2009.
[25] A. Gudipati and S. Katti. Strider: Automatic rate adaptation and collision handling. In *ACM SIGCOMM*, 2011.
[26] D. Halperin, T. Anderson, and D. Wetherall. Taking the sting out of carrier sense: Interference cancellation for wireless lans. In *ACM MobiCom*, 2008.
[27] Impinj Speedway. R420 RFID reader. www.impinj.com.
[28] E. Inc. Universal Software Radio Peripheral. http://ettus.com.
[29] T. Jayram and D. Woodruf. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with sub-constant error. In *SODA*, 2011.
[30] M. Karim and M. Sarraf. *W-CDMA and CDMA2000 for 3G mobile networks*. McGraw-Hill, 2002.
[31] D. Klair, K.-W. Chin, and R. Raad. A survey and tutorial of RFID anti-collision protocols. *IEEE Comm. Surveys*, 2010.
[32] M. Kodialam and T. Nandagopal. Fast and reliable estimation schemes in RFID systems. *MobiCom*, 2006.
[33] Y. Kou, S. Lin, and M. Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. *Transactions on Information Theory*, 2001.
[34] Laird Technologies. Crushcraft S9028PCRW RFID antenna. http://www.arcadianinc.com/.
[35] T. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge University Press, 1998.
[36] H.-C. Liu and J.-P. Ciou. Performance analysis of multi-carrier RFID systems. In *SPECTS*, 2009.
[37] C. Mutti and C. Floerkemeier. CDMA-based RFID systems in dense scenarios: Concepts and challenges. In *IEEE Int. Conf. RFID*, 2008.
[38] M. Pelino, C. Mines, J. Warner, and S. Musto. M2M connectivity helps telcos offset declining traditional services. Forrester Research, 2011.
[39] J. Perry, H. Balakrishnan, and D. Shah. Rateless spinal codes. In *HotNets-X*, 2011.
[40] E. Porat and M. Strauss. Sublinear time, measurement optimal, sparse recovery for all. In *SODA*, 2012.
[41] J. Posamentier. RFID tag clock synchronization. Patent US20070205871, 2006.
[42] PowerID. Battery assisted passive RFID tags read at 160+ feet. The RFID Network, 2012. www.rfid.net.
[43] D. Shen, G. Woo, A. Lippman, D. Reed, and J. Wang. Separation of multiple passive RFID signals using software defined radio. In *IEEE Int. Conference on RFID*, 2009.
[44] M. Sipser and D. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42:1710–1722, 1996.
[45] C. Swedberg. Visual data center combines RFID with 3-d thermal imaging. *RFID Journal*, July 2010.
[46] D. Tse and P. Vishwanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.
[47] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *ACM SIGCOMM*, 2009.
[48] A. Wang, S. Cho, C. Sodini, and A. Chandrakasan. Energy efficient modulation and mac for asymmetric RF microsensor systems. In *Int. Symposium on Low Power Electronics*, 2001.
[49] J. Yu, K. Liu, and G. Yan. A novel RFID anti-collision algorithm based on sdma. In *WiCOM*, 2008.
[50] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational RFID and sensing platform. Tech Report UMASS, 2011. http://spqr.cs.umass.edu/moo/.
[51] Y. Zhang, H.-H. Chen, and M. Guizani. *Cooperative Wireless Communications*. CRC Press, 2009.
[52] T. Zimmerman. Assessing the capabilities of RFID technologies. Gartner, 2009.