Laissez-Faire : Fully Asymmetric Backscatter Communication

Pan Hu, Pengyu Zhang, Deepak Ganesan
College of Information and Computer Sciences
University of Massachusetts, Amherst, MA 01003
{panhu, pyzhang, dganesan}@cs.umass.edu

ABSTRACT

Backscatter provides dual-benefits of energy harvesting and low-power communication, making it attractive to a broad class of wireless sensors. But the design of a protocol that enables extremely power-efficient radios for harvesting-based sensors as well as high-rate data transfer for data-rich sensors presents a conundrum. In this paper, we present a new *fully asymmetric* backscatter communication protocol where nodes blindly transmit data as and when they sense. This model enables fully flexible node designs, from extraordinarily power-efficient backscatter radios that consume barely a few micro-watts to high-throughput radios that can stream at hundreds of Kbps while consuming a paltry tens of micro-watts. The challenge, however, lies in decoding concurrent streams at the reader, which we achieve using a novel combination of time-domain separation of interleaved signal edges, and phase-domain separation of colliding transmissions. We provide an implementation of our protocol, LF-Backscatter, and show that it can achieve an order of magnitude or more improvement in throughput, latency and power over state-of-art alternatives.

CCS Concepts

•Networks → Network architectures; Wireless access networks;

Keywords

Backscatter; Wireless; Architecture

SIGCOMM '15, August 17 - 21, 2015, London, United Kingdom

DOI: http://dx.doi.org/10.1145/2785956.2787477

1. INTRODUCTION

As we enter a world where sensors are embedded in walls, wearables, and within bodies, a question that demands attention is what wireless technology is best suited for these devices. Backscatter has emerged as a strong contender for this regime because of its ability to deliver power while simultaneously offering an ultra-low power wireless backhaul.

One of the extraordinary benefits of backscatter is that it can help design wireless sensors that operate at end-to-end power budgets of under a few micro-watts, thereby enabling battery-less operation. For example, a backscatter-based temperature sensor that samples at 1Hz, and operates in a sense-transmit loop with no other overheads (i.e. no receive circuit, no protocol overhead, etc) would barely consume 10 μW of power, which makes it possible for such devices to operate continuously using a small amount of harvested power.

Backscatter is also attractive as a replacement to active radios on battery-powered sensors since it can support hundreds of Kbps while consuming only tens of micro-watts of power [26]. Backscatter achieves this power efficiency by shifting carrier generation to the reader, and only uses power for clocking its RF transistor. High-speed ultra low power radios can enable a paradigm shift in wireless sensing — continuous data offload from a variety of data-rich sensors such as cameras and microphones becomes extremely efficient, thereby enabling sophisticated distributed sensing applications.

While backscatter offers many advantages for wireless sensors, our ability to realize these benefits depends on the design choices made by the protocol. Seemingly innocuous protocol choices have important ramifications. For example, a protocol designed with the expectation that the radio can support bitrates of hundreds of Kbps significantly impacts power efficiency for simple low-rate sensors such as the backscatter-based temperature sensor described above. Despite its low sampling rate and limited communication needs, such a sensor would need to accumulate samples in a buffer, and use a high-speed clock to toggle its RF transistor, which increases power consumption by several tens of μWs over

a simpler design that used a slower clock and no packet buffers. This power difference hinders its ability to operate in a battery-less manner. On the other hand, optimizing the protocol for stringently constrained sensors by reducing the transmit and receive bitrate, would clearly hurt overall throughput. Even simple design choices like reader feedback to the tag (e.g. TDMA slot assignment messages [5]) incurs the cost of message decoding at the tag, which adds to the power budget. At its core, the issue is that protocols are often optimized for specific hardware capabilities, and it appears difficult to shoehorn a wide range of hardware scenarios into a protocol without sacrificing efficiency.

While dealing with hardware heterogeneity may appear to be an intractable problem, an examination of the capabilities at the reader presents intriguing possibilities. We have a powerful reader that can sample several orders of magnitude faster than the tags can modulate the signal (even the fastest sensor tags are barely going to exceed 100 Kbps). For example, take the case of a USRP-based EPC Gen 2 reader reading data from backscatter sensors — the reader can sample at 25 Mega samples/second, and even if a tag can transmit at 100 kbps, this means that less than 1% of the time-domain samples contain useful information.

In this paper, we argue that existing backscatter protocols do not take full advantage of its inherent asymmetry. Rather, these protocols make assumptions that often diminish the flexibility to design lower power or higher speed backscatter systems. Our line of attack is the following: we allow nodes to use the least restrictive backscatter protocol, a *laissez-faire* approach that lets nodes blindly transmit data once they see the carrier signal, and focus on the problem of decoding these concurrent streams by leveraging the more powerful reader. Consider the potential benefits of such a fully asymmetric design. One could envisage an extremely low-power tag that is virtually free of any computational logic — it senses and immediately transmits the digitized signal oblivious to any other wireless traffic. Such a design would need no decoding, no MAC, no packet buffers, and no high-speed RF oscillators. The model would benefit faster tags as well — a higher speed backscatter tag would not need to pause and wait for the slow tags to transmit their bits. But despite the intrinsic elegance of the laissez-faire approach, the pragmatics seem daunting. How can the reader cleanly separate the signals from different tags operating at widely different rates?

Our fundamental insight is that the laissez-faire approach is asynchronous, hence it results in temporal interleaving of signal edges caused by toggling of the transmit antenna at different nodes. By leveraging the reader's capability to sample at a much higher rate than the node, this time-domain asynchrony can be exploited to enable concurrent transfers. But temporal separation is insufficient on its own — edge transitions can collide since nodes independently decide when to transmit. To separate such collided signals, we take advantage of phase information about the backscattered signals. The backscatter baseband signal can be represented as comprised of a sine and a cosine wave. This provides us with a phase vector in the in-phase and quadrature dimensions (IQ vector). By leveraging the fact that collisions result in different clusters in the IQ plane, we can separate collided signals without requiring any special mechanism at the tag. The benefit of using time-domain separation followed by IQ-domain separation is important to emphasize — by itself, time-domain separation would not be able to resolve collisions, and IQ-domain separation would only support limited concurrency, but in tandem, they can enable substantial concurrency.

This paper is about the details of designing and implementing such a laissez-faire backscatter protocol. We ask and answer many questions to make the scheme practical — how can we separate signals across tags by leveraging the differences in signals across time, in-phase, and quadrature? what are the minimal restrictions we may want to impose to the laissez-faire model to ensure that the system actually works in practice? what are the performance benefits in terms of throughput, power, and latency? can the scheme support extremely constrained harvesting-based tags and higher rate battery-assisted tags?, and so on. Through this exploration, we hope to convince the reader that we have a system that retains much of the elegance and benefits of the laissez-faire approach, while imposing a few restrictions to make the technique practical.

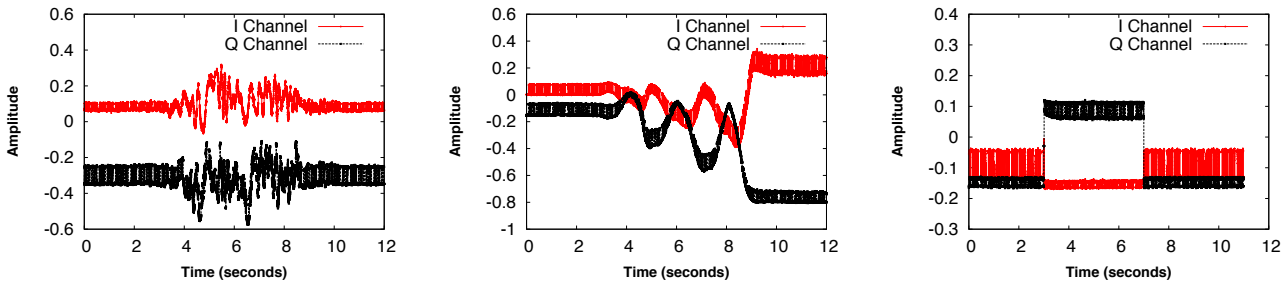
In summary, our key contributions are:

- We design a novel backscatter system, LF-Backscatter, that is fully asymmetric and enables flexible radio architectures at the tag, while pushing all decoding complexity to the reader. Our decoding algorithm leverages time, in-phase, and quadrature information to enable a high degree of concurrent transmissions from backscatter sensors. Our design has several novel contributions to extract edges robustly from interleaved signals, separate collisions, and correct errors, all without adding complexity at the tag.

- We build a full software and hardware prototype of LF-Backscatter, and evaluate its benefits in a range of scenarios. In a sixteen node experiment, we show that LF-Backscatter's throughput is 7.9× higher than Buzz and 16.4× higher than TDMA, its latency for reading identifiers is 9.5× faster than Buzz and 17× faster than TDMA, and its energy-efficiency is 20× lower than Buzz and two orders of magnitude lower than RFID chips [23].

2. CASE FOR LF-BACKSCATTER

In this section, we look at a variety of methods that have been proposed for supporting backscatter communication from multiple tags. We focus on backscatter



(a) People movement (b) Tag rotation (c) Coupled tags

Figure 1: Dynamics in received signal under different scenarios.

using Amplitude Shift Keying (ASK), which is the dominant backscattering technique since it provides continuous power to tags and enables relatively simple receiver design [8]. We start with sequential access, and work our way through previously proposed concurrent transmission methods, and finally discuss LF-Backscatter.

2.1 TDMA and CDMA-based approaches

Time Division Multiple Access (TDMA) is a natural choice for backscatter because the reader has the resources to co-ordinate transfer by estimating the number of tags, determining how many slots to use, and marking slot boundaries. Most backscatter-based systems are based on TDMA [7, 10, 25, 27], the most well known being EPC Gen 2 [5].

While TDMA is very well understood and easy to deploy, it is not efficient when considering heterogeneous tags with varying capabilities. If TDMA chooses to operate at a rate that is the lowest common denominator of the nodes in the network, then it would be ponderously slow and overall throughput would suffer. If TDMA chooses a faster rate that is high enough to offer better throughput, it forces tags to use more power-hungry components like faster clocks and larger buffers. TDMA is also less than ideal in that the reader issues periodic control messages, and nodes need to expend energy in decoding these messages. Finally, TDMA vastly underutilizes the sampling capability at the reader since it serializes transmissions.

Code Division Multiple Access (CDMA) is another approach for enabling concurrent transmission for backscatter sensors [12, 15, 17]. Here, each sensor exploits an orthogonal code for encoding data where each bit is converted to a long PN sequence (hundreds of bits in length). CDMA is inefficient for backscatter because data transmission takes a much longer time, which hurts throughput, or the tags need to operate higher frequency radios, which increases power consumption by orders of magnitude [22].

2.2 Linear Signal Separation

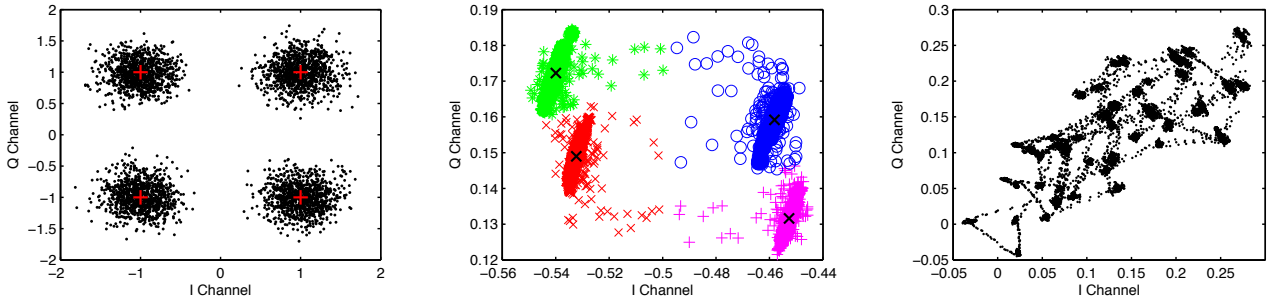
A recent concurrency-based backscatter protocol, Buzz [22], is based on the idea that backscatter signals combine linearly and therefore can be separated using matrix inversion methods. Buzz lets all nodes transmit in a synchronous bit-by-bit manner, so the received signal can be expressed as:

$$y = \mathbf{d}_{m\times n}\mathbf{h}_{n\times n}\mathbf{b}_{n\times 1} = \begin{pmatrix} d_{11} & \dots & d_{1n} \\ d_{21} & \dots & d_{2n} \\ \vdots & \ddots & \vdots \\ d_{m1} & \dots & d_{mn} \end{pmatrix} \begin{pmatrix} h_1 & 0 & \dots & 0 \\ 0 & h_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h_n \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad (1)$$

where y, the received symbol at the reader, is a linear combination of the complex channel coefficient corresponding to node i, h_i, the bit being transmitted by the node, b_i, and a randomization matrix, d_{ij}. Once the randomization matrix and the channel coefficients from each node are known, the function can be inverted to estimate the bits transmitted by each tag.

To implement this method in practice, Buzz first determines the channel coefficients of each node by using compressive sensing-based estimation. Once the channel coefficients are known, nodes transmit their bits in lock-step, re-transmitting each bit multiple times with different random combinations as determined by a pre-defined random matrix $\mathbf{d}_{m\times n}$. This allows the decoder to observe different combinations of the concurrent transmissions, enabling it to decode the stream. Once a combination with low error is determined, nodes move on to transmit the next message.

There are two problems that make this method less than ideal for heterogeneous backscatter sensors. One practical problem is that nodes need to transmit their bits in lock-step. This implies that the clocks on all nodes need to operate at the same rate, which is a reasonable assumption for RFIDs but not for sensors with heterogeneous capabilities and clock speeds. Another problem is that the channel coefficients need to be re-estimated to deal with changing environments and tag positions. Channel coefficients vary for three reasons. The first is when there is mobility of objects in the vicinity of the tag. In Figure 1(a), a tag is stationary in front of a reader while an individual moves around the room, resulting in substantial changes to the channel co-



(a) QAM IQ clusters. (b) Clustering of 2 tags (c) IQ clusters of 6 tags

Figure 2: IQ Clusters: (a) shows structured clusters in QAM, (b) and (c) show unstructured clusters. As the number of nodes increases, clusters are harder to separate.

efficients. Second, channel coefficients are also sensitive to movements of the tag. In Figure 1(b), a tag's orientation is varied by rotating it without displacing the tag, again resulting in significant changes to the channel coefficients. Third, channel coefficients also change when there is near-field coupling between the antennas of two or more tags. Figure 1(c) illustrates this case with a simple experiment where two tags were placed far apart, and then brought closer together. As shown, both channel coefficients are unchanged when the tags are about 1m apart, but when tags become closer together (roughly 5cm), there is near-field coupling across the antennas of the tags resulting in variations of channel coefficients. While such variations depend on the dynamics in the environment, adapting to them will incur protocol overhead for channel estimation prior to communication, and associated tag complexity for supporting the protocol functions. Buzz [22] does not explicitly address this problem since it targets one-shot RFID identification.

2.3 Cluster-based Separation of Signals

An alternative approach that has been considered in prior work is separation in the IQ plane [6]. When tags transmit simultaneously, their phase and amplitude information (IQ vector) creates multiple clusters, where each cluster corresponds to a specific combination of values from the tags. This approach could be viewed as the unstructured analogue of methods like Quadrature Amplitude Modulation (QAM) shown in Figure 2(a). While the signals in QAM are structured to be as far apart as possible, the clusters in this case are unstructured and depend on channel coefficients between each node and the reader.

For example, consider that there are two nodes that transmit simultaneously (lets call them 1 and 2) . Define $V(i, s_i)$ as the complex vector corresponding to node i when its antenna state is s_i (s_i is 0 when antenna is detuned and 1 when it is tuned). This vector is composed of the I (in phase) and Q (quadrature) channels, and can be expressed as $V(i, s_i) = I(i, s_i) + Q(i, s_i)$. Thus, the total signal reflected by the two nodes can be one of four options, $V(1, s_1) + V(2, s_2)$, where $s_1, s_2 \in \{0, 1\}$. Besides the signal reflected by the nodes, the reader also receives the signal reflected by the environment. For simplicity, let us assume that the reflection from the environment is a constant, so it won't affect the number of clusters, but will only add an offset to them.

Figure 2(b) shows the empirically obtained IQ constellation of received signal from the 2 nodes. We can see four dense clusters with sparse points between them. The sparse points are imperfect transitions between different states of transmitted signal.

A fundamental issue with this method is that it simply does not scale to a larger number of nodes. In the two nodes example, it is easy to see that simply choosing the closest cluster to a received vector can decode the signal from each node with high probability, but when we try to increase the number of tags, performance using this method degrades rapidly. This is because given N nodes (N $\geq$2), there are 2^N clusters in the IQ plot, resulting in clusters being closer to each other. An example with six nodes is shown in Figure 2(c). The figure has 64 clusters that are very close to each other, and dwell time in each cluster is short. This means that there are more points lie between clusters. In this case, separating the signal by classifying clusters is challenging. This drawback has been noted in prior work, for example, Angerer et al [6] also conclude that the technique does not scale to beyond two nodes.

2.4 Concurrency in Time and IQ

Our laissez-faire approach to backscatter is radically different from the above approaches. The underlying principle for signal separation is to leverage three axes — time, in-phase, and quadrature. When asynchronous ASK signals from tags are combined, two things happen — edges are created whenever there is a transition on any one tag, and the signals combine depending on the state of each tag as well as state of the environment. The underlying intuition in LF-Backscatter is that both

the edges and the combined signal are important for separating out the concurrent transmissions. Since the reader can oversample the received signal and edges are temporally localized, there is a lot of room to interleave edges within the time dimension of the combined signal.

As a concrete example, let us see how many UMass Moo tags, each transmitting at 100 Kbps, can be supported by a USRP reader sampling at 25 Mega-samples-per-second. This means that for each bit transmitted by a tag, the reader samples 250 times. An edge is roughly 3 samples wide at the reader's sampling rate, which means that we can stack $\frac{250}{3} = 83$ edges one after the other, so we can support a fairly large number of fast tags.

Of course, perfect interleaving is not achievable in practice, and we need methods to deal with the case where edges from different tags collide. But the fact that edges are finely localized in time means that only a small fraction of the concurrent transmitters are colliding at this edge. We take advantage of the temporal localization of collisions and make tags transmit periodically at multiples of a base-rate, which ensures that colliding nodes repeatedly do so at the same time, thereby giving us more information that can be used to separate the signals. While both linear separation and cluster-based separation are possible candidates given that we have a small number of colliders to separate, we use the latter since it does not require repeated estimation of the channel coefficients to each tag, and is less sensitive to changes in environment and orientation.

At a high level, the above description summarizes the main idea in this paper, but the devil is in the details. We have made several assumptions to narrate the high-level story — we assume that signal edges can be separated in a robust manner, that edges can be finely separated in time to maximize concurrency, that cluster-based separation can identify which bits are being sent by the colliding tags, that we can associate detected bits to the appropriate tag, and so on. We look at these thorny issues in the following section.

3. LF-Backscatter DESIGN

LF-Backscatter is designed to do as much as possible at the reader to ensure minimal complexity at the tag. We describe the sequence of stages involved in recovering data transmitted by the tags, starting with reliable edge detection.

3.1 Reliable edge detection

The first challenge that we face is reliably detecting signal edges, which is central to enabling concurrent laissez-faire transmission. It would seem that we can extract edges by using edge detection techniques on the combined signal received at the reader, but this turns out to be brittle.

Let us understand why this is the case. The backscatter reader sends a carrier wave toward the device, which can be represented as compromise of a sine wave and a cosine wave. These two parts are known as in-phase and quadrature signals. A backscatter device toggles its transistor and reflects this signal back to the reader. The amplitude of the signal is the total power of the received signal. To observe a significant change in amplitude when a node toggles its edge, the change in amplitude must be much higher than the background. However, the background is high when many other nodes are transmitting, and changes continually depending on the state of the tags. As a result, edges are not always clearly visible in this signal.

Our idea is to look at the IQ vector differential caused by the edge. Since the received signal from concurrent transmissions is a linear combination of individual signals (to first approximation), we can express the received backscatter signal at time t from N concurrent transmitters as:

$$S(t) = \sum_{j=1}^{N} S_I^j(t) + iS_Q^j(t) \tag{2}$$

where I and Q are the in-phase and quadrature channels respectively. To extract a signal edge, we look at the differential between the IQ signal before the edge occurs, and after the edge occurs i.e.

$$\Delta S(t) = S(t_+) - S(t_-) \tag{3}$$

where $S(t_+)$ and $S(t_-)$ are received signal by the reader at time t_+ after the edge and t_- before the edge. Thus, by subtracting the received signal after and before an edge, we can remove the effect of other background nodes. Since a single instance of t_+ and t_- could result in a noisy edge estimate, we use a set of points between the previous edge to the current edge as candidates for t_+, and a set of points between the current edge and the next edge as t_-, and take the average.

3.2 Separating edges into streams

At this point, we have a sequence of edges, but we do not know which node(s) toggled their transistors to create the edge. Each edge could be a) the result of a single node toggling its transistor, b) a collision resulting from multiple nodes doing so at the same time, or c) a spurious edge as a result of noise. To address this problem, we add some additional structure to the signal. First, the reader chops up time into shorter epochs, where each epoch is initiated by the reader by shutting off and re-starting its carrier wave. At the beginning of an epoch, each node picks a random initial offset, and begins to transmit its bits periodically at a selected rate starting at the offset. The randomization ensures that the collision pattern changes each epoch, so even if edges did collide in an epoch, they are likely to separate the next epoch. Second, we assume that the rate selected by the sensor is not arbitrary, but it is a multiple of a base rate (e.g. in our system, the

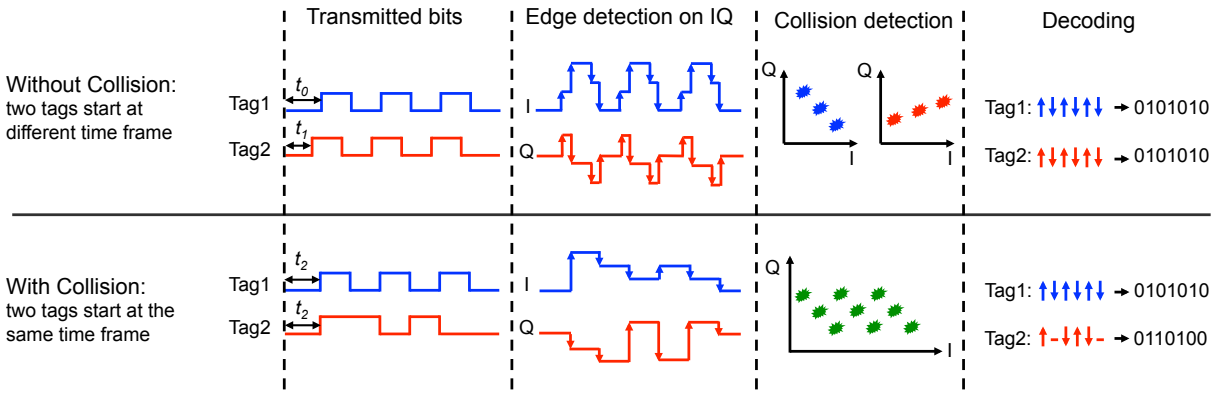


Figure 3: Figure shows processing pipeline of LF-Backscatter — on top, two nodes transmit concurrently but their edges do not collide, and on bottom, their edges collide. Time-domain separation extracts edges followed by cluster-based collision separation. When the nodes do not collide, edge differentials form three clusters in the IQ plane, whereas when edges collide, nine clusters are observed.

base rate is 100 bps, and any multiple of that is a valid data rate). This ensures that when nodes collide as a result of choosing the same offset, they continue to collide periodically throughout the epoch giving us more information to separate the signals. In addition, this assumption helps us detect if an edge is spurious, since such an edge would not have a repeating pattern at one of the valid rates.

Selecting fine-grained offsets One thorny problem that we face is how a node can choose a fine-grained offset when it only has a coarse-grained clock. Ideally, we want edges to be finely separated in time (from the reader's sampling perspective), so that we can support as much concurrency as possible. However, finely separated edges requires a fine-grained clock at the tag that is roughly as fast as the reader's clock, which defeats the very purpose of LF-Backscatter.

In keeping with the laissez-faire spirit, we just let tags start transmission the moment they see that the reader has turned on its carrier wave to signal the beginning of an epoch. Why would this work? To understand, let us look at the receive circuit of a backscatter tag. The energy from the incoming signal charges up a tiny receive capacitor, which in turn triggers a comparator when the voltage reaches a threshold that suggests that the carrier is turned on. There are three sources of randomness that affect exactly when the comparator fires: a) the energy received by the tag from the incoming signal, which depends on placement and orientation of the tag, b) the charging characteristics of the capacitor (typical capacitors have about 20% tolerance), and c) the noise in the charging process, since in practice the charging curve has small oscillations and isn't perfectly smooth as shown in textbook drawings. The end result is that tags exhibit natural variations in when they start their transfer, which gives sufficient temporal separation across edges.

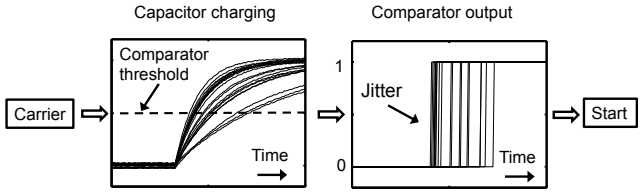


Figure 4: The capacitor charging curve changes depending on incoming energy, which in turn affects the time when the comparator fires to begin transmission.

Decoding edges Another question is how to robustly detect the presence of a stream and filter noise. To address this, the reader leverages the repeated pattern of edges, also referred to as an eye pattern [21]. Here, the analog value of a signal sample $s(t)$ is added to the analog signal sample that is T seconds ahead, $s(t+T)$, where T corresponds to the inverse of the highest bit rate. For example, T is 10μs for a 100kbps maximum bit rate. The eye pattern is determined for each possible offset, and used to detect the presence of a stream. The benefit of such folding is that it helps smooth out noise. Although each edge might have different strength $S+\sigma$, the noise σ will be averaged as a result of repetitive folding. Therefore, the peaks of eye pattern help explicitly identify edges that belongs to a same stream.

3.3 IQ Cluster-based Collision Detection

Once the streams are identified, the next step is to identify whether it is the result of a collision. To detect collided edges, we use the fact that the possible values of the edge differential vector depends on the number of nodes that are colliding at that position. Lets consider the example in Figure 3, and call the two transmitters as E_1 and E_2. For each node, edge differentials can be in one of three states (rising, falling, and constan-

Table 1: Single node data recovery. The first bit is the anchor, which helps disambiguate clusters.

Sent Bits	**1**	0	0	0	0	1	1	0	1	0	⋯
Received Edges		↓	-	-	-	↑	-	↓	↑	↓	⋯
Decoded Bits		1	0	0	0	0	1	1	0	1	0

t). In the top pipeline, time-domain separation unlinks the two streams, so each of them is clustered separately leading to three clusters on the IQ plot. On the bottom, we only have a single combined stream since the edges collide at the same time, so we now have nine states for the overlapped signal edge (each edge has three state, so there are nine combinations). Thus, when there are k nodes colliding, we expect 3^k clusters to be present. Given this intuition, we can detect if collisions are present by performing k-means clustering and determining the best fit in terms of number of clusters. If three clusters is not a good fit, then a collision is likely to have occurred.

We emphasize that it is not important to achieve highly scalable cluster separation since edges are finely localized in time, and a very small number of collisions are expected for a stream. Consider a simple case where nodes transmit at 100 Kbps, which is on the higher end of what we might expect in a real deployment. We use the same parameters as our experimental setup i.e. 16 nodes, 25Msps sampling rate at reader, and 3 sample edges. The probability of two-node collisions is 0.1890, whereas the probability of three node collisions is only 0.0181 making it far less likely. If the bit rate were lower, say 10 Kbps, the probability of three (or higher) node collisions is less than 0.0022 even when 200 nodes transmit concurrently.

3.4 Separating Collided Signals

While the different clusters can be separated using k-means, we need to map from cluster to the bit that each node transmits. Lets consider the case when a single node is transmitting, where an edge can have three states (rising, falling, constant). The cluster for the constant edge is located at the origin, so it is easy to determine. But the clusters for the other two edge states depend on the vector direction, which in turn depend on node placement, orientation, and other factors. The cluster to edge association is even harder when two nodes collide, since there are many more possibilities.

Let us first look at the case when only one node is present. Here, determining the sequence of bits only needs an *anchor* to tell us which cluster is +1. Since every epoch starts with a header from each tag, we embed a single anchor bit at a known location, which helps us disambiguate between the rising vs falling edge clusters. Once the cluster labels are known, the rest of the bits can be easily decoded. The process is shown in Table 1 where the first bit is an anchor with value one.

Lets turn to the case where two nodes collide. After correct classification, we have the centroid of nine

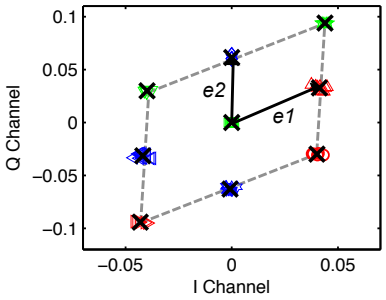


Figure 5: Nine clusters formed by two colliding edges. The clusters are linear combinations of the complex vectors corresponding to the edges.

clusters $C_1, C_2, ..., C_9$ where each centroid C_i maps to one of the nine pairs of states. Therefore, each centroid can be presented as a linear combination of signal edge vectors and direction coefficients as shown below:

$$\begin{aligned} a_1e_1 + b_1e_2 &= C_1 \\ a_2e_1 + b_2e_2 &= C_2 \\ &... \\ a_9e_1 + b_9e_2 &= C_9 \end{aligned} \tag{4}$$

where a_i and b_i indicate the direction of signal edges generated by these two devices ($a_i, b_i \in \{-1, 0, 1\}$), and e_1 and e_2 are the complex values corresponding to the edge differential. To associate each centroid with corresponding signal edges, we need to solve the above equations and obtain $a_1, a_2, ..., a_9$, $b_1, b_2, ..., b_9$, e_1, and e_2.

This problem is difficult to solve when a large number of nodes collide, but it is easily solved when we just have nine clusters. The cluster centered at the origin is easily decoded as constant from both nodes. From Figure 5, we see that the centroids of the remaining eight clusters form a parallelogram where the clusters in the mod-points of the edges correspond to the cases where only one of the edges is activated and and the other edge is a constant. Thus, if we identify the mid-points, we can find e_1 and e_2, and separate the edges into two streams. To identify the mid-points, we identify which groups of three cluster centroids are co-linear, which tells us which are the edge lines. We then select the mid-points of these co-linear centroids to identify the mid-point clusters, and thereby identify e_1 and e_2.

Note that one major advantage of our method in that it does not require estimation of the channel between the tag and reader. Our only assumption is that that channel coefficients are relatively stable during an epoch, which is reasonable since each epoch is a very short duration.

3.5 Bit Error Correction

The above process assumes that there are no missed edges or erroneously detected edges. If so, this can throw off the subsequent decoding, and cause errors in

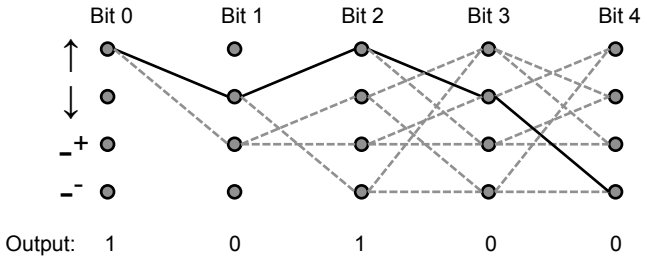


Figure 6: Viterbi decoder that encodes edge constraints. ↑ stands for a positive edge, and ↓ stands for a negative edge. $-^+$ means no edge found but its previous edge is a positive one and $-^-$ means its previous edge is negative

the rest of the sequence. While parity bits and error correction codes are commonly employed for error detection and correction, we would like to avoid having to compute anything at all on the tag-side since that increases complexity.

The fact that we are using signal edges to decode the sequence of ones and zeros transmitted by a tag suggests a possible solution. We simply leverage the fact that certain sequences are just not possible. For example, a rising edge followed by a rising edge is obviously an error. To correct for such errors, we use a Viterbi decoder with four states: ↑ (positive edge), ↓ (negative edge), $-^+$ (no edge found but previous edge is a positive one) and $-^-$ (no edge but previous edge is negative). We learn state transition probabilities, and calculate the emission probability by fitting the IQ values that are empirically observed to a two dimensional normal distribution: $(V_i, V_q) \sim N(\mu_i, \mu_q, \sigma_i, \sigma_q, r)$. We then use a Viterbi decoder to identify the most likely sequence of bits corresponding to the received sequence of edges.

Figure 6 shows an example where the bold line shows the decoded sequence of bits, and the dotted lines show alternate valid sequences. For example, if a node was transmitting a one as the first bit, then the only possibility is to either remain in the same state or see a negative edge (↓) following this state. The decoder leverages the valid state transition constraints together with analog information about the IQ vector length to correct errors that occur in the sequence of edges.

3.6 Tag Complexity

The steps that we have described so far require virtually no tag-side logic. We now describe two additions that could increase the complexity of tag design.

Clock: Since LF-Backscatter is primarily aimed at backscatter-based sensors that use a clock for sampling the sensor values, we can simply use the same clock for communication. LF-Backscatter clocks out bits as and when they are sampled and therefore does not need a separate clock for communication unlike typical wireless sensor platforms. However, since the decoding steps described above require tight timing for the signal edges in-order for the reader to decode the signals, we would need a clock with relatively low drift. If such a clock is not already available on a sensor platform, it can be added at the cost of a small increase to the overall power budget (e.g. the NXP PCF8523 Real Time Clock consumes $1.2\mu W$ [4]).

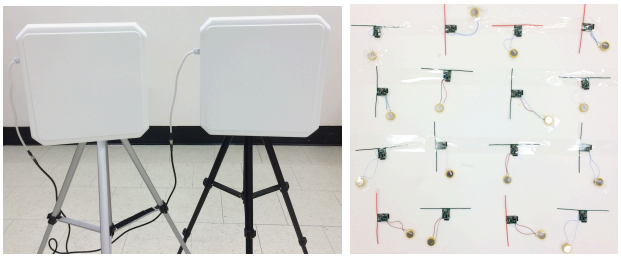


Figure 7: Experimental setup

Reliable data transfer: The design that we have described does not provide link-layer reliability to keep the tags simple. If link-layer reliability is needed, an acknowledgement mechanism can be added at the cost of some complexity at the tag. For example, a simple way to add reliability is for the the reader to send a Broadcast ACK to the entire network asking them to retransmit data for the next epoch. The benefit of this approach is that collision patterns are different across epochs, which can be used to decode messages. Similarly, the reader might broadcast a message to reduce the maximum bit-rate in the network to reduce collisions.

Note that stringently constrained tags can ignore these ACK messages. Since these tags typically transmit at a very slow rate, their transmissions are unlikely to cause collisions, so it is sufficient to slow down the faster nodes to reduce bit errors.

4. IMPLEMENTATION

In this section, we describe key implementation details not covered in earlier sections. We use the USRP N210 software defined radios and UMass Moo backscatter sensors for our instantiation of LF-Backscatter.

4.1 Platforms

The hardware platform is shown in figure 7.

USRP Reader LF-Backscatter is built based on the USRP N210 software radio reader developed by [24] with SBX RF daughterboards and Cushcraft 900 MHz antennas [1]. The SBX daughterboard has two RF interfaces where one (TX/RX) of them can be configured for either transmission or reception and the other (RX2) can only be configured as reception. The sampling rate at the reception antenna is set as 25MHz.

Backscatter node The UMass Moo is a backscatter node that operates in the 902MHz $\sim$ 928MHz band. Our decoding method can tolerate roughly 200ppm of clock drift, so we need to use an external low-drift crys-

tal oscillator rather than the built-in internal DCO on the Moo which has a typical drift of 40,000ppm (4% of the DCO clock speed [2]). The Moo has a low drift 32kHz external oscillator which we can use for decoding, but we replace this with a 8MHz oscillator to be able to experiment with higher speeds [3]. The clock we use has a typical drift of 150ppm.

4.2 Baselines

We compare LF-Backscatter with the following two baseline schemes:

- **TDMA:** We use a stripped down version of EPC Gen 2 where we remove a significant fraction of its protocol overhead (which is often substantial [10]), and keep the essential elements for TDMA operation. Following the EPC Gen 2 standard, we assume that slots are 96 bits long, and the bitrate is 100 kbps.
- **Buzz:** We also compare LF-Backscatter with Buzz [22]. Based on details in the paper, we reproduce the Buzz implementation including the tag-side code on the UMass Moo and reader-side decoding algorithms. The results from our implementation are comparable to numbers shown in [22], so we believe this is a faithful reproduction of their protocol. As with the TDMA scheme, each sensor transmits a 96-bit message each time at a bit rate of 100 kbps.

5. EVALUATION

Our goal in this evaluation is to demonstrate that LF-Backscatter can provide significant improvements in performance across three axes — throughput, latency, and energy.

5.1 Concurrent Transmission Goodput

Our first set of experiments looks at the benefits of LF-Backscatter in terms of throughout. We deploy sixteen backscatter tags at different locations roughly two meters from a backscatter reader, and select subsets of these tags for different experiments.

Aggregate throughput In the first experiment, we fix the bitrate of all the nodes to 100Kbps (since this rate can support most high-rate sensors), and increase the number of nodes from 4 to 16.

Figure 8 compares the aggregate throughput achieved by different schemes — LF-Backscatter achieves very close to the maximum possible throughput in all cases. The aggregate throughput is marginally lower than the maximum for the 12 and 16 node cases, but the difference is small. The throughput benefits are huge — when 16 nodes are present, LF-Backscatter is 16.4× better than TDMA and 7.9× better than Buzz.

Breaking down the benefits Let us now break down aggregate throughput by the different components of our design to see how much each of them matters.

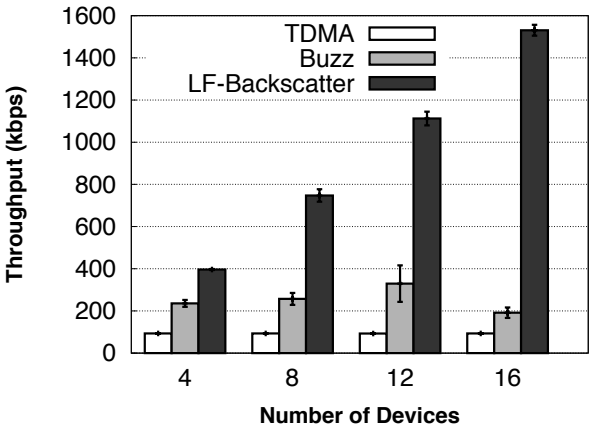


Figure 8: Comparison of throughput achieved by TDMA, Buzz, and LF-Backscatter when the number of nodes increases.

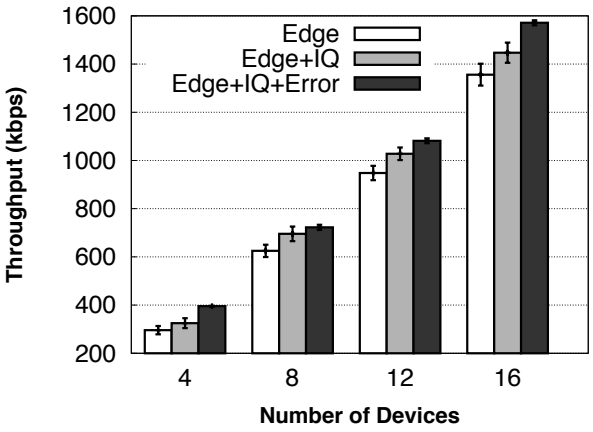


Figure 9: Breakdown of each decoding module's contribution to LF-Backscatter's throughput.

Figure 9 shows the breakdown — we start with an approach that uses only edge-based concurrency, then we add IQ cluster-based collision detection and separation, and finally we add error correction methods. We see that each of these benefits overall throughput — edge-based concurrency does really well by itself, but there's more error as the number of nodes increases.. For example, edge-based concurrency leaves about 15.3% of the throughput on the table for the 16 node case. Collision recovery improves throughput by about 5.6%, and adding error correction improves it by another 7.7%. Thus, all three design components play a crucial role in getting throughput to be close to the maximum.

Varying bitrate We can support sixteen nodes at 100 Kbps, but how far can we push it before we are unable to pack any more edges? Since we're limited in terms of number of available nodes, we instead vary the bit-rate of the sixteen nodes, and plot the aggregate throughput. Figure 10 shows that the aggregate

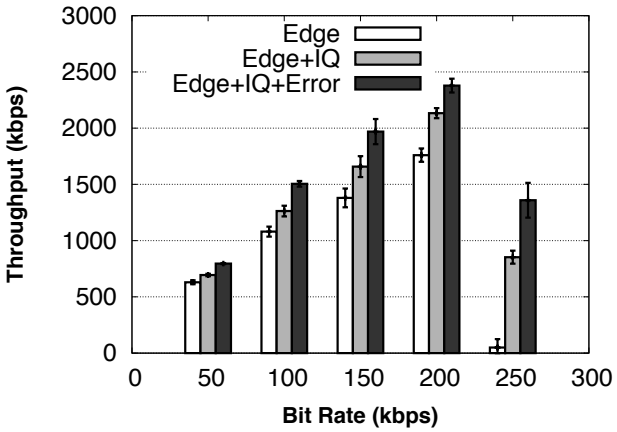


Figure 10: LF-Backscatter throughput when nodes transmit at different bit rates.

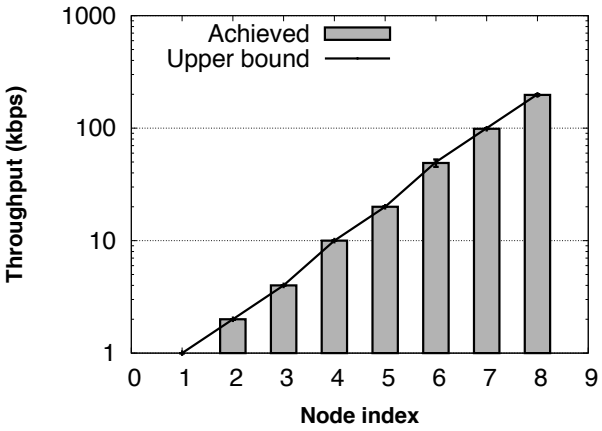


Figure 11: Throughput when a mix of nodes at different bitrates transfer concurrently. (y-axis in logscale)

Table 2: Separating edge collisions with IQ-based classification.

Setting	Accuracy
100 Kbps with background nodes	80.88%
100 Kbps w/o background nodes	86.89%
10 Kbps w/o background nodes	95.40%

throughput crashes after about 200 Kbps, so this gives us an empirical upper bound on the extent of concurrency we can support. The crash is not surprising. Our reader samples at 25 Mega samples/second, and tags transmit at 250 Kbps, so we can pack 33 nodes if their edges were stacked one after the other. Our deployment has 16 nodes, so there is a huge number of edge collisions. It is also interesting to note that the IQ cluster-based recovery and error correction are very useful as the bit-rate increases. Indeed, they pull throughput back to a respectable level at 250 Kbps even though practically all edges are colliding.

The result can be viewed from two lenses. From the perspective of wireless sensors, the result is very positive — concurrent transfer from 16 nodes at 200 Kbps is at least an order of magnitude better than what we can achieve from virtually all low power active radios in the market today, at a fraction of the power consumption. In addition, scalability to a few tens of nodes within a single communication range is often more than sufficient for typical sensor network or IoT deployments that currently rely on technologies like 802.15.4 or Bluetooth Low Energy. From the perspective of RFID deployments that target detection of hundreds or thousands of tags in a pallet, our solution is not ideal since our scalability is limited by our ability to interleave signal edges in the time-domain.

IQ cluster-based separation Let us take a closer look at IQ cluster-based separation, and see under what conditions it works and when it doesn't. Intuitively, our ability to separate depends on the relative distance between IQ vectors of the nodes that collide, which in turn depends on the placement of the nodes. To understand this, we place 16 nodes at different locations in front of a reader, and choose different pairs of nodes to transmit such that *all their edges collide*. We then try three cases: a) the two nodes transmit at 100Kbps, and there are 14 other nodes chattering away in the background, b) the two nodes transmit at 100Kbps, and there are no other tags transmitting in the background, and c) the two nodes transmit at 10Kbps, and there is no background transmissions. The three cases cover the effect of background as well as the effect of averaging noise (since edge decoding is more robust when edges are less frequent as described in §3)

Table 2 shows how well we can separate collided edges in different cases. Our classification accuracy is above 80% in all cases, with the lowest accuracy being in the case when there is a lot of background chatter. This is to be expected, since more nodes in the background increases the noise floor and results in a noisy edge differential, which in turn impacts accuracy. Removing background improves accuracy by 6%. Reducing the bit-rate improves accuracy by another 8.5% because we are able to average the edge differential over more samples (i.e. more samples before the edge and after the edge), so this helps improve SNR.

Co-existence of slow and fast tags So far, we have assumed that all tags are transmitting at the same rate but one of the key benefits of LF-Backscatter is that it can support widely different bitrates, ranging from ultra-low power tags to high-rate tags. To evaluate this benefit, we let two node transmit at each of the following eight sets of bitrates starting from slow to fast — 0.5 Kbps, 1 Kbps, 2 Kbps, 5 Kbps, 10 Kbps, 50 Kbps, and 100 Kbps.

Figure 11 shows the throughput achieved by each node. The results show that the slow nodes are not adversely impacted by the fast nodes, and have a loss rate of zero. This shows that LF-Backscatter supports sufficient concurrency that it can deal with nodes with widely varying data rates that we would typically expect from backscatter-based sensors.

5.2 Node Identification Time

While throughout is important in sensor data transfer scenarios, a canonical application of backscatter is to quickly read RFID identifiers in a shopping cart or warehouse. The performance metric of interest in this case is latency of reads. In general, higher throughout must mean lower latency for reading the identifier of tags, but one additional consideration is that we need to read identifiers in a reliable manner.

The node identification protocol that we build on LF-Backscatter works as follows. Each node transmits its EPC Gen 2 identifier (96 bits + 5 bit CRC) in each epoch with a random offset. Each tag starts at the highest bitrate (100Kbps), and at the end of the epoch, the reader can optionally send a command to use a lower bitrate if it observes too many collisions.

Figure 12 shows the identification time for different number of tags in front of a reader (up to 16). We see that the identification time is 17× lower than TDMA and 9.5× lower than Buzz, which means that RFID identification using our protocol can be considerably faster than existing techniques.

While we do not have hardware to experiment with more than 16 tags, we expect LF-Backscatter to scale quite easily to a larger number of tags. One easy approach is to set bitrate to a lower number, say 10kbps, and allow LF-Backscatter RFIDs to concurrently transmit their ID. In this setting, we can not only support a few hundred tags, we can also enable longer-range operation since the hardware is simpler than passive RFIDs (no decoding overhead, and much less complex circuit). We would also eliminate the achilles heel of TDMA-based RFID identification, which is the estimation of number of tags. It is well known that EPC Gen 2 has enormous overheads due to inexact cardinality estimation. While Buzz does not need cardinality estimation either, it assumes that channel coefficients of nodes are sufficiently distinct to separate, but this assumption may not hold as one scales the number of nodes.

5.3 Energy efficiency and tag complexity

Our third performance metric is energy-efficiency. It is hard to overstate the importance of energy-efficiency for harvesting-based sensors — operating range, throughput, latency, cost, deployability, size, and a variety of other metrics are inexorably connected with energy. The ability to design tags that operate at extremely low power is one of the key reasons to use LF-Backscatter.

To evaluate energy-efficiency, we first construct an FPGA-based implementation of LF-Backscatter so that we can fully understand its hardware complexity and measure power consumption. We implement LF-Backscatter and Buzz [22] in Verilog, and compare the number of transistors used by these protocols against a publicly available EPC Gen 2 implementation in Verilog [23]. All of our implementations currently support only a single bit-rate — while bit-rate adaptation can be added to all implementations, its overhead is similar across the three schemes, and would not affect the conclusions.

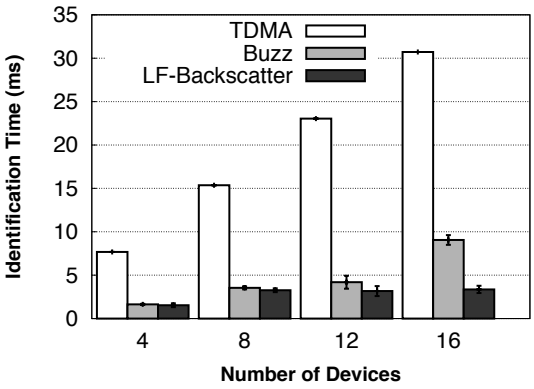


Figure 12: Node identification time when TDMA, Buzz, and LF-Backscatter are used for inventorying RFID tags.

Table 3: Hardware complexity of RFID chip, Buzz, and LF-Backscatter

Num of transistors	w/o FIFO	1k FIFO
RFID chip	22704	34992
Buzz	1792	14080
LF-Backscatter	176	176

Table 3 shows the results. EPC Gen 2 (TDMA) and Buzz need a packet buffer — the former to buffer sensor samples between transmission slots, and the latter to ensure that samples are not lost while bits are retransmitted in lock-step. This increases their complexity, so we provide results with and without the packet buffer. The results show a dramatic difference in hardware complexity — LF-Backscatter requires an order of magnitude fewer transistors than Buzz, and two orders of magnitude fewer transistors than EPC Gen 2, validating our claim that our protocol simplifies hardware design of tags.

Reduced hardware complexity typically implies lower power consumption, so lets look at the communication efficiency of the different protocol + hardware combinations. We obtain power numbers for the different hardware architectures from a SPICE simulation of our Verilog code, and we use throughputs from Figure 8 to calculate communication efficiency. Since the power consumption of EPC Gen 2 and Buzz depend on

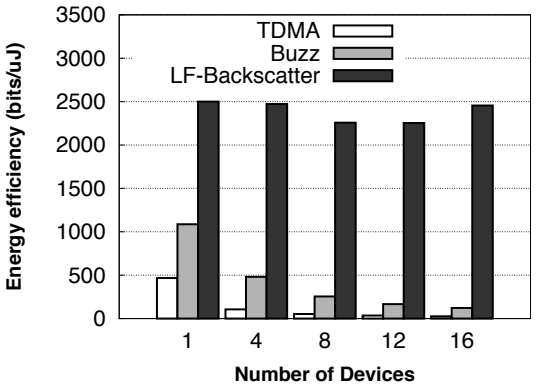


Figure 13: Energy efficiency (bits/μJ) of TDMA, Buzz, and LF-Backscatter.

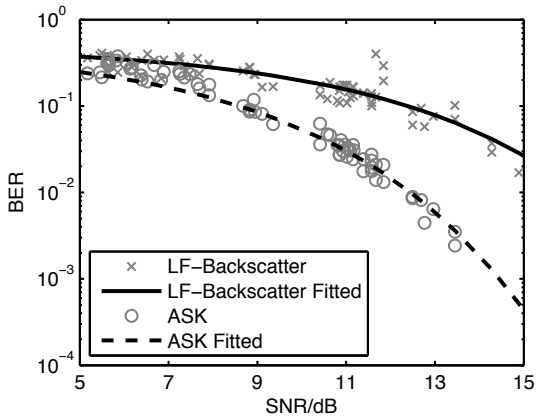


Figure 14: SNR vs BER of LF-Backscatter vs ASK. X-axis is cropped at SNR of 15dB after which BER goes to zero for both schemes.

the number of nodes that are transmitting, we vary the number of nodes from one to 16 and evaluate the power consumption for different approaches.

Figure 13 shows that LF-Backscatter is about 20× higher than Buzz and two orders of magnitude higher than EPC Gen 2 in terms of energy-efficiency. This benefit has huge implications — a 20× reduction in power consumption would mean roughly doubling the range at which the tags get sufficient energy from the reader, or alternately that 20× more data can be transmitted at the same distance. Either way, this is a big win.

5.4 SNR: LF-Backscatter vs ASK

LF-Backscatter has several benefits as described above, but trades off some robustness in the process since it relies on edge detection and requires higher SNR than ASK modulation. To evaluate the tradeoff, we compare the bit-error rates of the two decoding schemes for a single tag transmitting across many distances and reader power levels. We focus on the low SNR regime, which is where we might expect to observe less robustness for LF-Backscatter decoding. Figure 14 shows that there is a difference of approximately 4dB between LF-Backscatter and ASK in terms achieving equivalent bit error rates until the SNR reaches about 15dB, after which the bit error rate drops to zero.

To evaluate the impact of this SNR gap on range, lets consider the classical radar equation [19] used to determine backscatter link budget

$$P_r = P_t G_t^2 (\frac{\lambda}{4\pi d})^4 G_{tag}^2 K$$

where P_r is the received signal strength at the reader, P_t is the power of transmitted signal, G_t is the gain of transmit antenna, λ is the wavelength of received signal, d is the distance between transmit antenna and tag, G_{tag} is the gain of antenna at tag side, and K is the modulation loss of the tag. With this equation, we can calculate that if a tag has a working range of 10ft with ASK, it will have an equivalent range of 8.1ft with LF-Backscatter. Similarity, LF-Backscatter will have a working range of 23.7ft if a tag works 30ft with ASK. Thus, we expect LF-Backscatter to be effective for most of the working range of ASK modulation but at the higher end of the operating range, we need to switch from LF-Backscatter to ASK to improve robustness.

6. RELATED WORK

We discuss related work that we have not touched upon in the previous sections.

Concurrent transfer with active radios Much recent work has explored the problem of concurrent wireless communication with active radios such as WiFi [11, 9], typically using Successive Interference Cancelation (SIC). However, SIC requires significant difference in the SNR level of the collided signals, hence it only applies to limited scenarios [18]. Methods like ZigZag [9] may, however, be useful if we use reliable transfer and nodes repeat the same bits in different epochs.

Alternate modulation schemes While backscatter is predominantly based on ASK modulation which is the basis for our work, other methods like Frequency Shift Keying (FSK) or Quadrature Amplitude Modulation (QAM) are also possible [20]. FSK is less efficient than ASK since it requires multiple edge transitions for each bit, so the energy efficiency and throughput of LF-Backscatter is certainly better. QAM could have similar throughput but it is certain to involve considerably more complex hardware at the tag (consequently more power), as well as more hardware complexity at the reader (consequently more expensive).

Fit with other backscatter systems There have been many exciting new ideas for backscatter-based applications including backscatter of TV or FM signals for credit card transactions [16], gesture recognition using backscatter [14], backscatter with WiFi routers [13], etc. While mileage may vary depending on extent of concurrency that is needed, several of these cases can leverage

the ideas in LF-Backscatter to reduce tag complexity, increase throughput, improve energy-efficiency, etc.

EPC Gen 2 Much work on backscatter has focused on addressing issues with EPC Gen 2 (e.g. [10, 24, 25, 27]). While EPC Gen 2 is here to stay, we think it shouldn't limit us from exploring new designs. Indeed, LF-Backscatter allows considerably more efficient tag designs, and lower latencies than EPC Gen 2, so we think there is much to be gained from this exploration.

7. CONCLUSION

In this paper, we introduce LF-Backscatter, a novel asymmetric communication paradigm for backscatter that pushes all the complexity to the reader, and enables tags to communicate virtually without restriction. We think that this is an important design point in backscatter communication, and the laissez-faire approach at the tag has significant ramifications in terms of designing lower power and perhaps lower cost backscatter sensors, while simultaneously being able to support high-speed concurrent backscatter communication. Our experimental results show benefits in many way that backscatter may be used, whether for high-throughput transfer, low-latency reads, or highly energy-efficient tags, typically by an order of magnitude or more over existing schemes.

Acknowledgement

We thank our shepherd Dina Katabi for providing guidance in preparing our final draft and the anonymous reviewers for their insightful comments. This research was partially funded by NSF grants CNS-1218586, CNS-1217606, CNS-1239341, and NIH grant 1U54EB020404.

8. REFERENCES

[1] Laird Technologies. Crushcraft S9028PCRW RFID antenna. http://www.arcadianinc.com/.

[2] MSP430 Mixed Signal Microcontroller. http://www.ti.com/lit/ds/symlink/msp430f2418.pdf.

[3] NX3225GD: 8MHz crystal clock. http://www.ndk.com/images/products/catalog/c_NX3225GD-STD-CRA-3_e.pdf.

[4] PCF8523 real-time clock (RTC) with calendar. http://www.nxp.com/documents/data_sheet/PCF8523.pdf.

[5] Radio-Frequency Identity Protocols Class-1 Generation-2. http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.

[6] C. Angerer, R. Langwieser, and M. Rupp. Rfid reader receivers for physical layer collision recovery. *Communications, IEEE Transactions on*, 58(12):3526–3537, 2010.

[7] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: An energy-aware runtime for computational rfid. In *NSDI*, 2011.

[8] K. Finkenzeller. Rfid handbook: Fundamentals and applications in contactless smart cards and identification, 2003.

[9] S. Gollakota and D. Katabi. *Zigzag decoding: combating hidden terminals in wireless networks*, volume 38. ACM, 2008.

[10] J. Gummeson, P. Zhang, and D. Ganesan. Flit: a bulk transmission protocol for rfid-scale sensors. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 71–84. ACM, 2012.

[11] D. Halperin, T. Anderson, and D. Wetherall. Taking the sting out of carrier sense: interference cancellation for wireless lans. In *MobiCom 2008*, pages 339–350. ACM, 2008.

[12] L. Kang, K. Wu, J. Zhang, H. Tan, and L. M. Ni. Ddc: A novel scheme to directly decode the collisions in uhf rfid systems. *Parallel and Distributed Systems, IEEE Transactions on*, 23(2):263–270, 2012.

[13] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall. Wi-fi backscatter: internet connectivity for rf-powered devices. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 607–618. ACM, 2014.

[14] B. Kellogg, V. Talla, and S. Gollakota. Bringing gesture recognition to all devices. In *Usenix NSDI*, volume 14, 2014.

[15] L. Kong, L. He, Y. Gu, M.-Y. Wu, and T. He. A parallel identification protocol for rfid systems. *Infocom*, 2014.

[16] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith. Ambient backscatter: wireless communication out of thin air. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 39–50. ACM, 2013.

[17] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith. Turbocharging ambient backscatter communication. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 619–630. ACM, 2014.

[18] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi. Successive interference cancellation: a back-of-the-envelope perspective. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 17. ACM, 2010.

[19] M. I. Skolnik. Introduction to radar. *Radar Handbook*, 2, 1962.

[20] S. J. Thomas and M. S. Reynolds. A 96 mbit/sec, 15.5 pj/bit 16-qam modulator for uhf backscatter communication. In *RFID 2012*, pages 185–190. IEEE, 2012.

[21] D. Tse and P. Viswanath. *Fundamentals of wireless communication*. Cambridge university press, 2005.

[22] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk. Efficient and reliable low-power backscatter networks. *ACM SIGCOMM*, 42(4):61–72, 2012.

[23] D. Yeager, F. Zhang, A. Zarrasvand, N. T. George, T. Daniel, and B. P. Otis. A 9 a, addressable gen2 sensor tag for biosignal acquisition. *Solid-State Circuits, IEEE Journal of*, 45(10):2198–2209, 2010.

[24] P. Zhang and D. Ganesan. Enabling bit-by-bit backscatter communication in severe energy harvesting environments. In *USENIX NSDI 2014*.

[25] P. Zhang, J. Gummeson, and D. Ganesan. Blink: a high throughput link layer for backscatter communication. In *MobiSys'10*, pages 99–112. ACM, 2012.

[26] P. Zhang, P. Hu, V. Kumar, and D. Ganesan. Ekhonet: High speed ultra low-power backscatter for next generation sensors. In *ACM Mobicom 2014*.

[27] Y. Zheng and M. Li. Read bulk data from computational rfids. *IEEE Infocom*, 2014.