Leveraging Interleaved Signal Edges for Concurrent Backscatter

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

ABSTRACT

One of the central challenges in backscatter is how to enable concurrent transmissions. Most backscatter protocols operate in a sequential TDMA-like manner due to the fact that most nodes cannot overhear each other's transmissions, which is detrimental for throughout and energy consumption. Recent efforts to separate concurrent signals by inverting a system of linear equations is also problematic due to varying channel coefficients caused by system and environmental dynamics. In this paper, we introduce BST, a novel physical layer for backscatter communication that enables concurrent transmission by leveraging intra-bit multiplexing of OOK signals from multiple tags. The key idea underlying BST is that the reader can sample at considerably higher rates than the tags, hence it can extract time-domain signal edges that result from interleaved transmissions of several tags. Our preliminary experiment results show that BST can achieve 5× the throughput of Buzz and 10× the throughput of TDMA-based solutions, such as EPC Gen 2.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords

Concurrent, Backscatter, Signal Processing

1. INTRODUCTION

Backscatter communication has seen a revival in popularity in recent years due to the potential to enable ultra-low power data transfer between sensor tags and infrastructure. There has been substantial recent work on backscatter, including efforts to improve range [9], improve throughput [7, 10], leverage different power harvesting sources [6, 3], and enable new applications [5, 8].

One of the central challenges in backscatter is how to enable concurrency. Most protocols designed for backscatter operate in a sequential TDMA-like manner due to the fact that nodes cannot overhear each other's transmissions [1]. This approach has several performance drawbacks. First, the overall network-wide bandwidth is limited by the maximum rate at which individual tags can transmit, often only a few tens or hundreds of kilobits/second. Second, control messages from the reader for enabling TDMA scheduling has both bandwidth and power implications. In terms of bandwidth, control messages are slow since the tags are generally expected to be resource-limited and unable to decode data at a fast rate. For example, the reader to tag data rate in EPC Gen 2 is only 40∼160 kbps, whereas the tag to reader data rate is up to 640kbps. Into terms of power, control messages require all tags to be listening and processing messages, which has high energy overhead on passively powered tags (74% of the power of running a backscatter radio [4]). In principle, concurrent transfer can address these drawbacks. By letting nodes transfer concurrently, the aggregate data rate can be higher than a single tag, and control overhead can be amortized over many concurrent transmissions.

But enabling concurrency in backscatter is challenging. Backscatter tags are typically highly constrained, and incapable of leveraging computationally sophisticated multiple access techniques such as CDMA. Further, since tags do not actively generate carrier wave, FDMA cannot be achieved on backscatter tags. Finally, any approach that requires more sophisticated transmission circuitry at the tag comes at the cost of higher power consumption, which defeats the backscatter power advantage. Thus, concurrency has to be enabled while retaining the underlying simplicity and low power nature of backscatter.

In recent years, there have been a few approaches suggested for achieving such concurrency in backscatter. One interesting approach is Buzz [7], which leverages the fact that the received signal is a *linear combination* of the complex channel coefficients from each tag to the reader, and the bits being transmitted from each tag. Thus, the signals can be decoded once the channel coefficients to each node are learnt. While this approach can be effective in some deployments, we argue that channel coefficients are not as stable and predictable as one might expect, which makes decoding challenging. Another approach from Angerer et al [2] is to use the received phase and amplitude information to create multiple clusters for identifying the collided bits, where each cluster corresponds to a specific combination of bits from the nodes. This approach works well when there are only two or perhaps three concurrent nodes. However, its performance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
HotWireless'14, September 11, 2014, Maui, Hawaii, USA.
Copyright 2014 ACM 978-1-4503-3076-3/14/09 ...$15.00.
http://dx.doi.org/10.1145/2643614.2643617.

degrades significantly when number of nodes increases because the number of clusters increases exponentially.

In this paper, we introduce BST[1], a physical layer technique for backscatter networks that enables concurrent transmission from multiple devices by leveraging temporally interleaved signal edges of the OOK signals from multiple tags. The key idea underlying BST is that the reader can sample at considerably higher rates than the tags, hence it can extract time-domain signal edges that result from interleaved transmissions of several tags. Since nodes transmit via OOK, edges carry important information regarding the bits being transmitted, hence allow us to decode data from different tags.

Our contributions are three-fold. We carefully investigate the drawbacks of previous techniques that have been proposed for concurrency in backscatter. We then describe the central ideas in BST, and how we can reliably detect signal edges and leverage them to decode interleaved streams from multiple tags. Finally, we develop an algorithm for dealing with collisions between edges. We present a preliminary implementation of BST on a USRP based backscatter reader and UMass Moo platforms and show that the throughput achieved by BST is 5× higher than Buzz [7] and 10× higher than TDMA-based solutions, such as EPC Gen 2 [1].

2. CASE FOR BST

In this section, we argue that existing approaches for collision recovery in backscatter have significant flaws that prevent them from scaling to larger number of nodes and achieving higher throughput.

2.1 Vector based Collision Recovery

One approach from Angerer et al [2]. is to leverage the fact that 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 nodes. This approach is similar to methods like Quadrature Amplitude Modulation (QAM) shown in Figure 1(a), but the major difference is that while the signals in QAM are structured to be as far apart as possible, the clusters in our case are unstructured and depend on channel coefficients between each node and the reader.

For example, consider that there are two tags that transmit simultaneously. In the I (in phase) channel, the signal of each tag reflected when sending 0 is $I_{(i,0)}$ and $I_{(i,1)}$ when sending 1. Similarly in the Q (quadrant) channel the signal is $Q_{(i,0)}$ and $Q_{(i,1)}$ respectively. Define complex vector V:

$$V_{(i,0)} = I_{(i,0)} + Q_{(i,0)} \tag{1}$$

$$V_{(i,1)} = I_{(i,1)} + Q_{(i,1)} \tag{2}$$

The total signal reflected by both tags can be one of four options (depending on the bit, s_i, transmitted by each tag):

$$\begin{pmatrix} \Sigma V_1 = V_{(1,0)} + V_{(2,0)} & , s_1 = 0, s_2 = 0; \\ \Sigma V_2 = V_{(1,0)} + V_{(2,1)} & , s_1 = 0, s_2 = 1; \\ \Sigma V_3 = V_{(1,1)} + V_{(2,0)} & , s_1 = 1, s_2 = 0; \\ \Sigma V_4 = V_{(1,1)} + V_{(2,1)} & , s_1 = 1, s_2 = 1; \end{pmatrix} \tag{3}$$

[1]BST stands for Backscatter Spike Train, representing a sequence of edges in time.

Besides the signal reflected by the tags, 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 1(b) shows the empirically obtained IQ constellation of received signal generated by 2 tags. We can see four dense clusters with sparse points between them. The sparse points are imperfect transitions between different states of transmitted signal.

Lack of scalability: 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 tags (N >2), there are 2^N clusters in the IQ plot, resulting in clusters being closer to each other. An example with six tags is shown in Figure 1(c). The figure has 64 clusters that are very close to each other, and dwell time in the cluster is short, which means there are more points lie between clusters. In this case, separating the signal by using spatial-division multiplexing is very difficult [2].

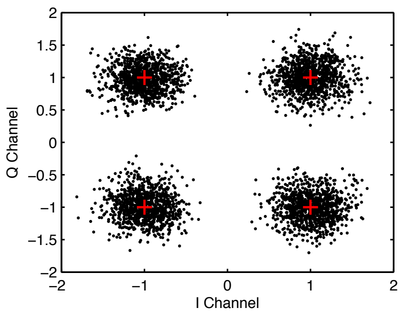
2.2 Signal Inversion for Collision Recovery

A second approach for decoding concurrent transfers from tags is to leverage the fact that backscatter signals are narrowband, and the received signal is a *linear combination* of a) the complex channel coefficients from each tag to the reader, and b) the bit being transmitted from each tag. This can be expressed as: $y = \mathbf{h}_{1\times n}\mathbf{b}_{n\times 1}$, where y, the received symbol at the reader, is a linear combination of the complex channel coefficient corresponding to node i, h_i, and the bit being transmitted by the node, b_i. Once the channel coefficients from each band are known, the function can be inverted to estimate the bits transmitted by each tag.

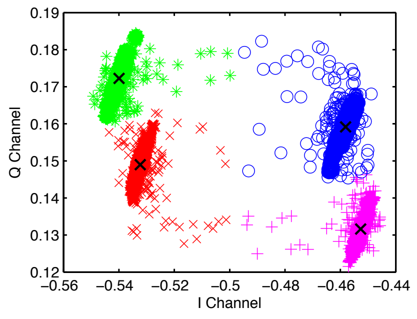
Buzz [7] is a protocol that leverages this idea. Briefly, Buzz first determines the channel coefficients of each node by using a compressive sensing method. Once the channel coefficients are known, nodes transmit their message in a synchronized manner with slot boundaries being aligned. To allow the reader to decode which node is transmitting which bit, the nodes re-transmit the same bit multiple times with different random combinations as determined by a predefined random matrix. This allows the decoder to observe different combinations of the concurrent transmissions, enabling it to decode using a belief propagation algorithm that continuously searches for the lowest error combination. Once a combination with low error is determined, nodes move on to transmit the next message.

A key problem with this approach is that the channel coefficients need to be known a priori in-order for the scheme to work, which makes it unsuitable in scenarios where either the node or the environment changes frequently.

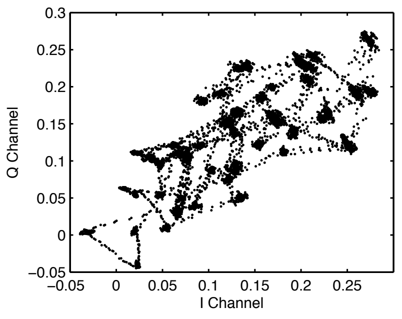
Channel coefficients can change for three reasons. The first reason why channel coefficients can change is when there is mobility of objects in the vicinity of the tag. In Figure 2(a), a node is stationary in front of a reader while an individual moves around the room, resulting in substantial changes to the channel coefficients. Second, the channel coefficients is also sensitive to even small movements to the tag. In Figure 2(b), a node's orientation is varied by rotating it without displacing the node, again resulting in significant changes to the channel coefficients. Third, channel coeffi-



(a) 4QAM IQ clusters.



(b) Clustering of 2 tags



(c) IQ plot of signal generated by 6 tags

Figure 1: 4QAM, IQ plot with 2 tags, and IQ plot with 6 tags.

cients also change when there is near-field coupling between the antennas of two or more tags. Figure 2(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 nodes are about 1m apart, but when nodes become closer together (roughly 5cm), there is near-field coupling across the antennas of the nodes resulting in variations of channel coefficients. To make matters worse, the coupling depends on the bit transmitted by each tag, making decoding extremely hard. Buzz does not explicitly address this problem, but dealing with variations in channel coefficients is costly since the compressive sensing-based estimation process is complex and elaborate.

3. BST DESIGN

The core primitive in BST is reliable edge detection. We discuss this primitive, and then show that we can design a reliable concurrent transfer protocol over this layer.

3.1 Leveraging interleaved signal edges

Backscatter is an asymmetric communication system in that the reader is far more powerful than the tag. Thus, the reader sampling rates are often orders of magnitude higher than the maximum rate at which any single tag can transmit. For example, the USRP reader can sample at 100 million samples per second, while a typical Moo sensor can transmit at a maximum of 250 kbps. The implication is that even if nodes transmit in an interleaved manner, resulting in many more edges than a single node can generate, the reader can oversample to detect these edges as long as there is a small amount of temporal separation across the edges. Of course, the implicit assumption that we make is that the edges are separable and don't overlap, and this is a key consideration that we address in this paper.

While edges are not particularly useful for complex encoding or modulation schemes such as OFDM, backscatter devices are simple and use OOK modulation. As a result, each edge carries information about when the tag toggled its transistor. This information, in turn, can be leveraged to decode the signal as we will soon describe.

Different from systems discussed in section 2 that depend on the combination of signal and channel condition, BST relies on edges. Clearly, edges are not impacted by changes in the channel coefficients unlike alternate methods. Also, if different nodes have different SNRs, then edges corresponding to nodes with strong SNR will be decoded with fewer errors, and will have fewer retransmissions. In terms of scalability, edge-based techniques scale much better than use of IQ clusters; however, as we show later, collisions can increase when too many nodes interleave their edges.

We now turn to a more detailed description of BST. The decoding process of BST includes two steps: time-domain edge detection and bits interpretation from detected edges. We start with reliable edge detection.

3.2 Reliable edge detection

Before diving into practical approaches for detecting signal edges, it is useful to understand the structure of the ASK (OOK) signal generated by backscatter devices.

Signal model: When a backscatter reader communicates with backscatter devices, its carrier wave is sent on both I and Q channels. Therefore, the reflected signal generated by toggling a transistor also have two components: one reflection on I channel and the other on the Q channel. For a single transmitter, the received signal at the reader is:

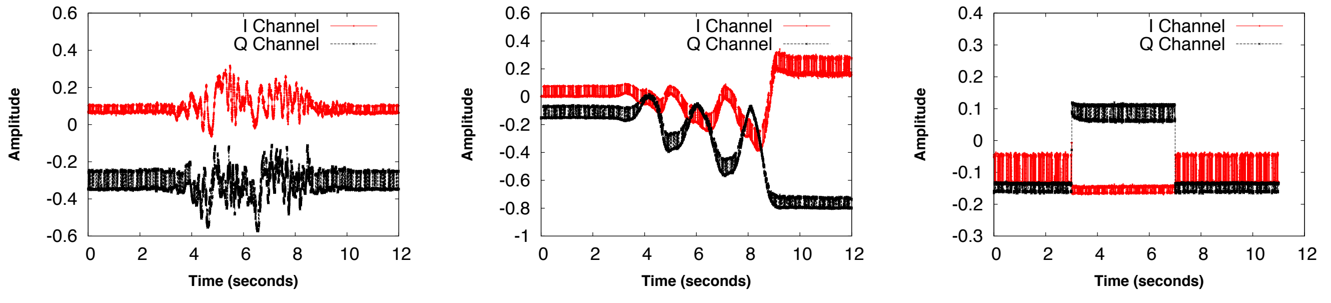
$$A_j = s_j V_{(j,I)} + \bar{s_j} V_{(j,Q)} \tag{4}$$

In this model, s_j is the reflection coefficient, and $V_{(j,I)}$ and $V_{(j,Q)}$ are signal vectors on I and Q channels. When multiple backscatter devices transmit to a reader simultaneously, their signal linearly add up on I and Q channels. Therefore, for concurrent transmission, the received signal at the reader is:

$$\Sigma A = \sum_{j=1}^{N} s_j V_{(j,I)} + \sum_{j=1}^{N} \bar{s_j} V_{(j,Q)} \tag{5}$$

Edge detection: A key observation is that the linearly combined signal received by a single reader still contains the edge information of each individual backscatter transmitter. This is a result of the linear signal addition described in the previous signal model. When an individual backscatter device toggles its transistor, it introduces edges on both I and Q channels of the combined signal at the receiver, which enables the receiver to detect the edge.

Figure 3 shows an example of how edge detection can be done on I and Q channels. Let's assume that only one backscatter device toggles its transistor in this example. The background signal, which are generated by other backscatter devices as well as the signal reflected by ambient environmental background, is represented by a signal vector $V(bg)$.



(a) Received signal with people moving (b) Received signal with tag rotating (c) Received signal with coupling

Figure 2: Dynamics in Channel Coefficients

When the device transmits, it generates a reflected signal $V(tx0)$ for data zero and $V(tx1)$ for data one. The signal received by a backscatter reader is $V(rx0)$ and $V(rx1)$ respectively, which are the addition between background signal and transmitted signal. Intuitively, a single device's signal edge detection can be done by checking the difference between $V(rx0)$ and $V(rx1)$ as the following equation shows:

$$\Delta A = |\overrightarrow{V_{diff}}| = |\overrightarrow{V(rx1)} - \overrightarrow{V(rx0)}| \tag{6}$$

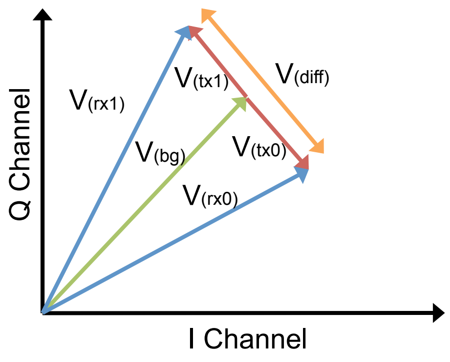


Figure 3: Edge detection based on signal vectors on I and Q channels

When the differential vector is greater than a predefined threshold, an edge of a single device is detected. Some approaches, such as EPC Gen 2, use amplitude of the signal for edge detection. However, those approaches lose phase information and lead to reduced SNR as well as potential errors in detection.

3.3 Assigning edges to nodes

Once edges have been detected, the next question is how to assign edges to the corresponding nodes that generated them. Assigning an edge to a node without any priori information of transmitters is difficult. Therefore, we make each node transmit in a periodic manner with a pre-defined period. This is shown in Figure 4 where the first node starts transmission at t_0 and the second starts at t_1. Those two nodes use different periods, T_1 and T_2, for transmitting information. Because of the difference of starting time as well as transmission period, the reader can then separate the different sequences of edges by looking for the streams that have different time-offsets.

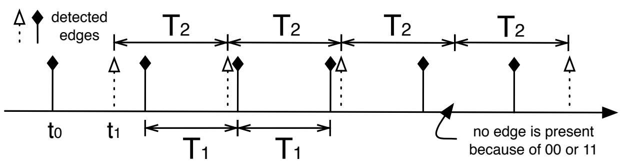


Figure 4: Detected edges of several transmitters

Once the streams are identified, the next stage is to identify gaps in the edge sequence — these gaps are not errors, instead, they carry critical information about instances where a "00" or "11" occurs (hence the lack of an edge). They can be identified by detecting the presence of an edge at time $t_i + nT_i$ where t_i is the starting time of transmission and T_i is the period of transmission. Thus, we now have a sequence of ones and zeros, where ones correspond to a change in the transmission symbol (one to zero or vice-versa) and zero corresponds to no change in the transmission symbol.

3.4 Recovering data from edges

Once we know the edge sequence corresponding to each node, we can now figure out the actual bit stream being transmitted if we have an anchor that tells us the starting point. For example, let's say that we know each stream starts with a one. Then, given the sequence of edges, we know at what time points the value changed from 1 to 0, or vice-versa, hence we can decode the bitstream from the edges. The process is shown in table 1. The data starts with 1 and the correct output is decoded bits 2.

Of course, this 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 the rest of the sequence. To avoid this, we insert a sentinel bit at specific intervals. For example, consider the case where we insert a "1" after every byte in the data. Then we know that every 9th bit should be a 1 in the decoding, which lets us bootstrap the decoding of the subsequent byte, and lets us detect single edge errors in the previous byte.

Table 1: Data Recovery

Sent Bits	1	0	0	0	0	1	1	0
Relative Bits	x	1	0	0	0	1	0	1
Decoded Bits 1	0	1	1	1	1	0	0	1
Decoded Bits 2	1	0	0	0	0	1	1	0

3.5 Handling edge collisions

Our discussion so far assumes that edges are not overlapping and therefore easily separable. The obvious question is how often edges from different nodes overlap, and how to deal with this problem.

Let us first ask how frequently edges overlap. Set the baud rate for backscatter node to 100kbps, so this is the rate at which an individual node generates edges. In our current implementation, the edge detection resolution is about 3 samples at 25M samples per second from the reader. Thus, if two edges were to appear within 3 samples of each other, it would result in a collision.

We use two methods to deal with edge collisions. The first approach that the reader tries to locally perturb colliders to avoid collisions. To achieve this, the reader sending a short *jitter* pulse when it detects a collision, causing the nodes that just transmitted to re-try with a new offset. If this method fails, the second approach that the reader tries is to asks nodes to reduce their bit rate by half if the number of retries exceed a certain threshold. For example, for 16 concurrent transmitters with a bit rate of 100 kbps, the collision probability is 0.93. which means an expectation of 13.1 retries before success in the start-up process. However, if we reduce the bit rate by half to 50 kbps, collision probability drops to 0.72, which means only 3.6 retries are needed before a success.

4. PRELIMINARY RESULTS

We now present some initial results for BST. Figure 5 shows the aggregate communication throughput achieved across multiple concurrent transmitters for a fixed bit rate. In this experiment, we deploy 1, 2, 4, 8, 12, and 16 backscatter devices 1 meter from a backscatter reader. As the number of transmitters increases, the obtained aggregate throughput at the backscatter reader also increases. We observe a linear increase when there are fewer than twelve transmitters, which means that the reader is able to successfully decode all the edges from these transmitters. When the number of transmitters is larger than twelve, the aggregate throughput degrades as a result of collided edges — as described earlier, this would need to be addressed by reducing the bit rate.

As baseline, we show the throughput of a TDMA based scheduling algorithm, which has a roughly fixed throughput irrespective of the number of tags. The throughput of BST is up to 10× higher than a TDMA-based approach as the number of transmitters increases.

5. CONCLUSION

In this paper, we introduce BST, a novel physical layer technique for backscatter networks that enables concurrent transmission from multiple devices. The key idea in BST is to leverage high-rate sampling backscatter reader and detect interleaved signal edges to decode collided bits from multiple concurrent transmitters. We propose an algorithm for reliable signal edge detection and discuss how to use these edges to decode interleaved streams from multiple tags. In the case of a dense tag deployment where signal edges might collide with each other, we also develop an algorithm for resolving collisions between edges. Our experimental results show that BST can achieve 5× to 10× throughput improvement over existing approaches.

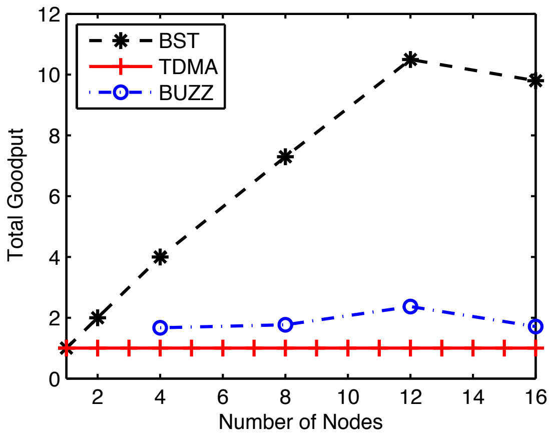


Figure 5: BST Goodput

Acknowledgement

We thank the anonymous reviewers for their insightful comments. This research was partially funded by NSF grants CNS-1218586, CNS-1217606, CNS-1239341.

6. REFERENCES

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

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

[3] J. Gummeson, S. S. Clark, K. Fu, and D. Ganesan. On the limits of effective hybrid micro-energy harvesting on mobile crfid sensors. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 195–208. ACM, 2010.

[4] 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.

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

[6] 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.

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

[8] 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.

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

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