EkhoNet: High Speed Ultra Low-power Backscatter for Next Generation Sensors

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

ABSTRACT

This paper argues for a clean-slate redesign of wireless sensor systems to take advantage of the extremely low power consumption of backscatter communication and emerging ultra-low power sensor modalities. We make the case that existing sensing architectures incur substantial overhead for a variety of computational blocks between the sensor and RF front end — while these overheads were negligible on platforms where communication was expensive, they become the bottleneck on backscatter-based systems and increase power consumption while limiting throughput. We present a radically new design that is minimalist, yet efficient, and designed to operate end-to-end at tens of μWs while enabling high-data rate backscatter at rates upwards of many hundreds of Kbps. In addition, we demonstrate a complex reader-driven MAC layer that jointly considers energy, channel conditions, data utility, and platform constraints to enable network-wide throughput optimizations. We instantiate this architecture on a custom FPGA-based platform connected to microphones, and show that the platform consumes 73× lower power and has 12.5× higher throughput than existing backscatter-based sensing platforms.

Categories and Subject Descriptors

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

Keywords

Architecture, Backscatter, Sensor, Wireless

1. INTRODUCTION

A fundamental assumption that has driven the design of sensor networks for decades is that communication is the most power-hungry component of an individual sensor system. The power consumption gap between communication and other modules has driven a plethora of design choices in sensor networks, primarily by encouraging designers to reduce data at the source, thereby minimizing the amount of data that needs to be communicated.

MobiCom'14, September 7-11, 2014, Maui, Hawaii, USA.

http://dx.doi.org/10.1145/2639108.2639138.

Table 1: Power consumption of accelerometer, audio, ecg, and image sensors.

	Accel [1]	Audio [2]	ECG [3]	Camera [14]
Power	6μW	15.3μW	60μW	0.7μW

We argue that this assumption does not hold when it comes to passive radios such as backscatter. Backscatter requires extraordinarily simple circuitry since the carrier wave is generated by a reader, and a sensor only needs to modulate the signal to transmit information, thereby eschewing power-hungry components of a typical active radio. The simplicity and inherent efficiency of backscatter means that the energy gap between communication and other components of a system has narrowed dramatically.

These observations have profound implications on the design of next-generation wireless sensing systems that operate using backscatter. The primary implication is that the bottleneck in terms of power consumption has shifted away from communication to computation and sensing. But sensing is often not the bottleneck as well — the past decade has seen dramatic reductions in the power consumption of sensors such as microphones, cameras, ECG, accelerometers, and others, many of which consume only μWs of power while sampling at high rates (Table 1). Thus, both backscatter communication and a variety of low-power sensors can operate at μWs of power, and the key question becomes one of optimizing the rest of the system to match these numbers. This requires that we re-think every component between the sensor and RF interface — data acquisition, data processing, buffering, packetizing, MAC, and many others now become the bottleneck for achieving ultra-low power operation.

In this paper, we overturn the design principle governing wireless sensor design from one that is focused on minimizing communication to one focused on optimizing the computational elements between the sensor and RF interface. But optimizing computation is easier said than done, and requires an understanding of every module of the sensing platform, in-depth analysis of how to eliminate overhead from these modules, and design of a modified architecture to support an optimized design.

But our efforts to optimize computation raises an unexpected problem. If we do nothing to reduce data at the source, we need the bandwidth to be able to transfer raw data from the sensor to infrastructure. While backscatter communication is efficient in terms of power, throughputs achieved by practical backscatter-based systems have been

abysmal. Despite several efforts at improving throughputs of backscatter [13, 27, 8, 22, 12], the best case throughput is still only around 20 kbps even when only a single node is present, and drops dramatically to barely hundreds of bits/second when there are multiple devices sharing the network. These numbers are not encouraging — for example, a microphone sampled at 8-44 KHz requires transmit rates upwards of 704 kbps, a far cry from the throughput that backscatter platforms are able to support today.

This leads us to the central question that we address in this paper: *how can we design a backscatter-based wireless sensor system that achieves whole-system power consumption of μWs, while simultaneously increasing data rates to support raw data transfer from sensors at several hundreds of kilobits/second.* Our goal is aggressive — as a point of comparison, an existing backscatter-based sensor, the UMass Moo (or the UW WISP) consumes about 2mW of power while transmitting at a few kilobits/second when there are multiple devices present. Thus, we seek to drop the system-wide power consumption by more than two orders of magnitude while simultaneously enabling two orders of magnitude increase in the data rates.

Our contributions are two-fold. First, we present a novel backscatter-based sensor platform, Ekho, that achieves our design goal to optimize power by eliminating computational overhead from the sensor to RF pipeline. We start with a deep dive at what computational modules are present between the sensor and RF interface on a typical low-power sensor platform, and measure their power consumption, before launching into a minimalist design that is optimized for power. Our second contribution is a network stack, EkhoNet, that is designed to be minimalist and enable bandwidth scale up to support data rates of hundreds of Kbps while supporting tens of nodes. While each Ekho node is minimalist, our MAC layer leverages resources at the reader to enable utility-energy and channel-aware optimization of bit rates and slot sizes across nodes.

Our results on a USRP reader and Ekho nodes show that:

- For operating an accelerometer at 400Hz, Ekho consumes 35μW of power, 7.6× lower than the 266μW of the Moo and 3.3× lower than the 118μW of WISP5.0. For operating an audio sensor at 44kHz, Ekho consumes 37μW of power, 76× lower than the Moo and 13.5× lower than the WISP5.0.
- We show that EkhoNet can scale to a network of several high bandwidth sensors. When a network of ten Ekho nodes equipped with microphones transmit simultaneously to a reader, we achieve a throughput of 780 kbps as a result of interleaving the data streams at the MAC layer. We also use an energy-utility-channel aware scheduler, and show that over 50% of the audio sensors achieve a median MOS score larger than 2, significantly higher than a baseline scheme that assigns sampling rates evenly across all nodes.

2. CASE FOR Ekho

In this section, we make the case that backscatter communication is extremely cheap and overturns the widely held premise that communication is more expensive than computation. We focus on the tradeoff between computation and communication since many commonly used sensors are already extremely efficient in terms of power. We begin with a discussion of why backscatter is efficient.

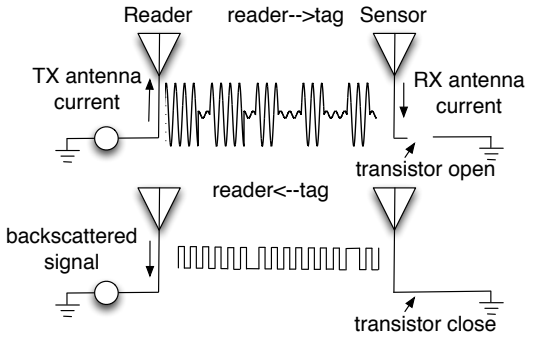


Figure 1: Backscatter communication basics.

2.1 Backscatter radio RF front end

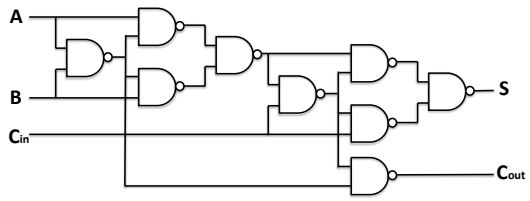
Backscatter radios are designed to enable ultra low power wireless communication. As shown in Figure 1, a reader provides a carrier wave, which can be modulated with information to enable ultra low power wireless communication. While the carrier wave can also be rectified by a sensor for energy harvesting, our focus in this paper is on backscatter as a low-power radio, whether energy is obtained via harvesting or a battery, hence we focus on the communication rather than harvesting aspects of backscatter.

To transmit data, a sensor toggles the state of a transistor to detune its antenna and reflect the carrier wave back to the reader with its own information bits. Because the sensor does not actively generate RF signal as active radio systems, the power consumption of the backscatter radio is very low. In addition, the on-off transition overhead of backscatter radios is very short because backscatter radios do not have to warm up the RF analog circuits for data transmission unlike active radio systems. As a result, there is little overhead incurred while transmitting via backscatter, even when transmitting at a high rate. For example, one key component of the backscatter analog RF front end of the WISP [6] is a MOSFET transistor (BF1212WR). Its power consumption follows the equation of CV^2F where C is the capacitance of the transistor, V is the digital drain-source voltage, and F is the frequency of operating the transistor. When this transistor is toggled at a slow rate of 10Hz, it consumes 55pW of power, and even when toggled at a high rate of 1MHz, it only consumes 5.5μW of power. Thus, backscatter radios consume of the order of μWs of power, even for high rate data transfer.

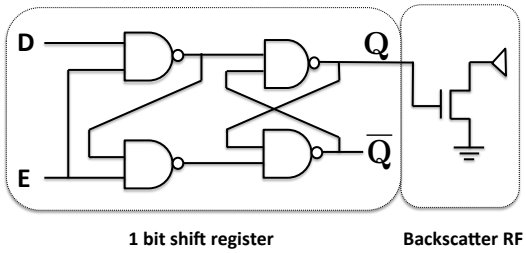
2.2 Why compute if its cheaper to transmit?

The power consumption of backscatter radio has surprising implications on sensor system design, and challenges long-held views about communication vs computation tradeoffs in these systems.

Computation vs Communication: A common assumption in designing sensor systems has been that computation is significantly cheaper than communication, often by many orders of magnitude. This view has shaped a plethora of efforts for in-network processing, signal compression, subsampling, and other such approaches to reduce data at the source prior to communication. Indeed, this tradeoff has been reinforced by performance/power trends over the past decade — power consumption of embedded processors have



(a) 1 bit adder circuits for computation.



(b) 1 bit shift register controls a backscatter transistor.

Figure 2: 1 bit adder and 1 bit shift register circuit.

dropped dramatically, while power reduction in active radios has been relatively slower.

However, backscatter communication challenges this long-held view. Backscatter is inherently extraordinarily efficient since the carrier wave is generated by the reader, and the tag only backscatters the signal without any additional amplification. Thus, each bit of backscatter is extremely simple, and only requires a handful of gates (Figure 2). This implies that for computation to be cheaper energy-wise, the computational operations on each bit would have to use fewer gates than that required to communicate the bit. This is often a tall order due to the simplicity of backscatter.

Consider, for example, a simple aggregation operation that sums ten sensor readings before transmitting the aggregate value over the radio. On traditional sensor platforms, such data reduction would have direct and significant power benefits since communication dominates power, and our aggregation scheme cuts this cost by a factor of ten. The same operation on a backscatter-based platform has dubious benefits. Figure 2 shows that the number of NAND gates required for summing two bits is roughly nine (thirty six transistors), but only four NAND gates (sixteen transistors) and an additional transistor for backscattering the signal are needed to transmit the same data via the shift-register controlled backscatter RF! As power consumption is proportional to number of transistors, a nine gate adder consumes 2.1× more power than the shift-register controlled backscatter RF.

It is necessary to add a few caveats to our simplified comparison of computation and communication. The clock rates of communication subsystems are limited by signal to noise ratio considerations, whereas the clock rates of processors can be higher, and thereby reduce power. In addition, low-power processors use many tricks to reduce power consumption including optimized signal processing circuits, different power domains, extremely tight duty-cycling, and so on. Despite these optimizations, the cards are stacked against computation. Backscatter is so incredibly simple in terms of circuitry that even matching the efficiency of backscatter becomes a challenging architectural design problem.

Thus, the crux of our argument is the following: *backscatter drives down the optimal cross-over point between computation vs communication, such that communication of raw data may be preferable to computation in a wider spectrum of real-world scenarios.*

Implications on architecture design: This observation has an immediate implication on the architecture of a backscatter-based sensor platform. Traditional sensing platforms add a lot of computational modules between the sensor and the radio for sensor data acquisition, processing, filtering, buffering, etc. The contribution of these components to overall power consumption of an active radio-based sensor system is minimal and can largely be ignored. However, on backscatter-based platforms, these components become the bottleneck.

This raises an intriguing question — with the power consumption of backscatter being so low, would it in fact be more efficient to eliminate all of these modules en-bloc, and just connect the sensor directly to the radio? In other words, would it be better to just stream every bit of data that is sensed directly through the radio?

We take a measurement-driven approach towards answering these questions. First, we look at the computational blocks between sensing and the RF interface on existing backscatter-based sensing platforms to understand how much power they consume, as well as why they suffer in terms of throughput. Second, we build on our empirical study and design a radically new backscatter-based sensor platform that addresses these limitations.

3. INVESTIGATING EXISTING WIRELESS SENSING ARCHITECTURES

In this section, we investigate why current backscatter-based platforms are unable to achieve end-to-end power consumption of μWs for high-rate sensing and transfer. We also investigate why they are unable to achieve high-data rate communication, particularly while operating at low power. To empirically understand these factors, we look at the UMass Moo/UW WISP class platforms that are equipped with sensors, a low-power MCU (MSP 430 family) and a backscatter radio.

3.1 Poor energy efficiency

We start with a break down of the power consumed by three key computational modules on a UMass Moo (Figure 3): 1) the sensor data acquisition subsystem which handles the protocols for operating sensors, 2) the data handling subsystem on a micro-controller where sensor data is stored, processed (if needed), formatted into packet, and sent to the network stack, and 3) the network stack implemented in a combination of hardware and software.

3.1.1 Sensor data acquisition

Sensor data acquisition is a relatively simple operation — some sensors have an on-board ADC, hence data acquisition is via a protocol such as SPI or I2C, whereas other sensors just provide an analog signal which is digitized using the micro-controller's ADC. Despite its simplicity, even these operations are not as cheap as one might expect. For example, sampling an accelerometer via the SPI bus would require

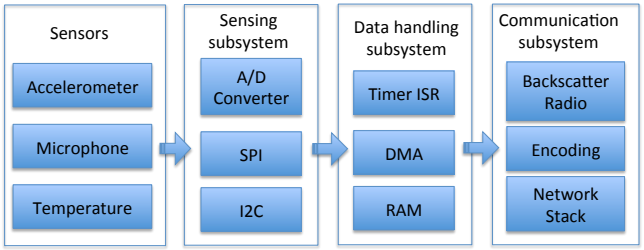


Figure 3: Computational blocks on existing backscatter-based sensors.

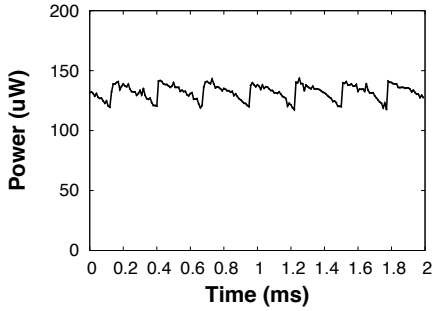


Figure 4: Power consumed for handling timer interrupts at 4kHz. The MCU is unable to switch to sleep mode due to frequent interrupts.

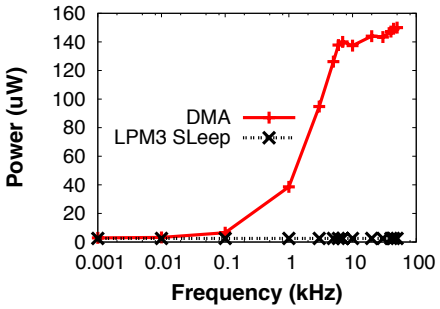


Figure 5: The power consumption of DMA transfer at different frequencies.

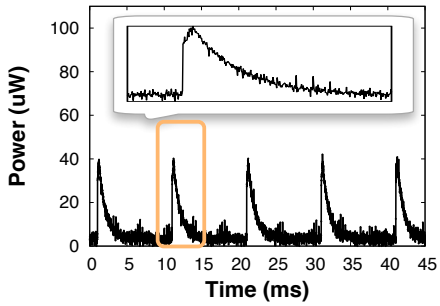


Figure 6: Power consumption of DMA transfer at 100Hz. DMA is slow to return to sleep mode.

periodic wakeup of the MCU to fill the SPI buffer, sending the read command and read address to the sensor, as well as providing the clock for the SPI bus. The overall result is that the MCU is active for about 40% of the time when acquiring data from an accelerometer sampling at 400 Hz. This acquisition operation, in itself, consumes 84μW of power, 14× higher than the accelerometer (6μW). The cost of acquiring audio data is equally high — when sampling an audio sensor (ADMP803) at 44KHz, acquisition consumes 492μW of power, 14.5× higher than the audio sensor (34μW).

3.1.2 Data handling subsystem

The data handling subsystem is the block that processes the acquired sensor data, formats and packetizes it, and sends it to the network stack. To minimize this overhead, sensor systems typically operate in a duty-cycled mode where the MCU is turned on for a minimal amount of time needed to handle the data, before switching back into sleep mode to conserve energy.

However, this optimization is no longer effective when this subsystem handles high-rate sensors. Figure 4 shows the power consumption for executing the timer interrupt service routine to handle each acquired audio sample. At high rate, the MCU is rarely able to switch completely back into the ultra-low power sleep mode due to frequent interrupts. Thus, the overall power consumption of the data handling module is roughly the ballpark of active mode power consumption of the MCU (a few mW), which is several orders of magnitude higher than the power consumed by the sensor.

One method to reduce power of the data handling subsystem is to use Direct Memory Access (DMA), which allows transfer of data from the sensor to memory without waking up the MCU. This raises the possibility that waking up the MCU can, perhaps, be avoided altogether if the data is transferred directly from the sensor to the network queue without any processing.

Surprisingly, DMA does not reduce power consumption. Figure 5 shows empirically measured power consumption for DMA transfer on an MSP 430, which moves the sensor data from a sensor to a local memory at different frequencies. We observe that while DMA is efficient at low rates (e..g below 100Hz), it has high power consumption at high transfer rates — for example, DMA transfer consumes 149.2 μW of power at 44 kHz, 60× higher than the 2.5 μW of LPM3 sleep mode of the MCU. This is surprising since one would expect that the MCU is in sleep mode while DMA operates.

The culprit for high power consumption of DMA turns out to be its tail energy consumption. Figure 6 shows the power consumption of repeated DMA transfer at 100 Hz. This experiment is done with an MSP 430 set to LPM3 sleep mode and a timer that periodically triggers DMA transfer. When a DMA transfer is initiated, its power consumption increases to 40μW within 10us, and starts decreasing once the DMA transfer is done. However, the power consumption decays at a relatively slow rate compared to the sharp increase, resulting in a long tail of roughly 3.5ms. When the DMA transfer frequency is high, such as 5kHz shown in Figure 5, the long tail leads to high power consumption. While we are not certain about the cause of this behavior, one hypothesis is

that the system waits for more data before it times out and switches to a lower power mode. This behavior is common in many power savings circuits, for example, in smartphone radios [7, 17], and is typically done to amortize the cost of waking up and shutting down a hardware subsystem.

3.1.3 Communication subsystem

The final computational component of a sensor platform is the communication stack, which includes the PHY, MAC and upper layers. While the RF interface of backscatter is extremely low-power, the other layers add more overhead. For example, on the UW WISP or UMass Moo platforms, the backscatter radio is controlled by a hardware timer which needs to be configured and handled in software. In addition, the EPC Gen 2 network stack on these devices is implemented in software, and results in substantial overhead since the MCU needs to handle protocol messages. In fact, the MCU needs to be on for 67% of the time for processing network stack messages at the software layer while only 7% of the time is used for data transmission. As a consequence, the software on UMass Moo platform consumes 2mW of power, which is three orders of magnitude higher than the power consumption of a low-power sensor.

As with the data handling subsystem, the software overhead of the network stack can be reduced by using hardware peripherals to control the radio. One commonly available hardware peripheral on MCUs is the Universal Asynchronous Receiver and Transmitter (UART). This is particularly useful for a backscatter radio since UART generates an ASK signal, which can be directly transmitted via backscatter (which uses OOK). At the first glance, the UART peripheral has the potential to dramatically reduce the cost of running the network stack because it can operate when the MCU stays in deep sleep mode. However, its buffer needs to be filled with sensor data, which in turn needs to be done with either DMA or software, both of which are expensive energy-wise. As a result, even the UART-driven backscatter radio consumes roughly 2mW.

3.2 Poor transmission efficiency

The second key drawback of existing backscatter-based sensors is the abysmal throughputs that they achieve. For example, even though there have been many efforts to improve backscatter throughput, the ceiling is still less than 20kbps for a single node [13, 27, 8, 22], and drops to hundreds of bits/second in a network with multiple devices. Clearly, this is far below what is needed for streaming raw sensor data from high-rate sensors.

One factor that limits the throughput is the poor efficiency in clock utilization. For example, the UMass Moo and WISP take 48 clock ticks to send a single bit of data, which causes a 48× reduction of the maximum possible throughput that is achievable with the system clock. We find three reasons for this inefficiency. First, both transmission and reception logic is implemented in software which, naturally, is inefficient in the use of the clock. Although the transmission and reception code on the Moo and WISP platforms are optimized in assembly instructions, one bit transmission and reception still has substantial overhead. Second, EPC Gen 2 PHY-layer encoding further reduces the clock utilization efficiency. To minimize the DC components during data transmission, each bit is encoded into a sequence of pulses using Miller encoding. For example, the Miller-4 encoding used by Moo and WISP platforms uses eight pulses to encode one bit of data, resulting in further drop of throughput by a factor of eight. Third, the EPC Gen 2 MAC layer is extremely inefficient for high bandwidth data transfer. While this is a point that has been made many times before [13, 27, 8, 22], an efficient alternative that achieves high throughput using backscatter is lacking.

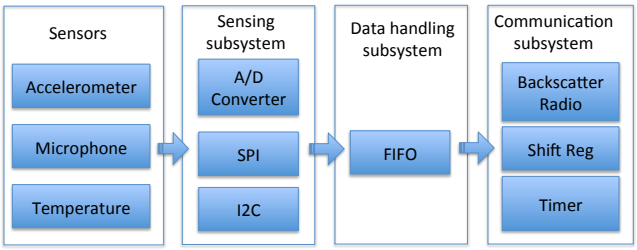


Figure 7: The key components of Ekho.

3.3 Summary

Thus, the limitations of the computational blocks on existing backscatter-based sensor platforms lead us to the following observation. The primary culprit in terms of power is the MCU's active mode power consumption, and the fact that many operations (sensor acquisition, data handling, communication) require execution of instructions on the MCU. Surprisingly, optimizing the system by leveraging hardware peripherals such as DMA and UART do not solve the problem, particularly at high data rates due to tail power consumption, and coupling between different components of the sensing to communication pipeline. In terms of throughput, the primary issues stem from inefficient utilization of the clock due to a combination of software overheads, encoding overheads, and an inefficient MAC layer standard. In conjunction, these limitations call for a *clean-slate re-design of a backscatter-based sensor platform* from the ground up for extremely low power consumption and high data rates.

4. THE Ekho PLATFORM

Our solution is Ekho, a backscatter-based sensor platform that is optimized for ultra low power operation and high-speed streaming from sensors. We outline the platform architecture followed by the MAC layer.

4.1 Eliminating computational blocks

At the platform level, the design of Ekho is minimalist. We simply remove as many computational blocks between the sensor and RF analog front end as possible in favor of communicating raw data. Figure 7 shows the key components in Ekho.

Ekho reduces the overhead of data acquisition from the sensor by implementing the SPI and ADC sampling logic on a small CPLD (FPGA). Implementing these blocks in hardware means that we can make them as fast as needed without incurring the software overhead of waking up a microcontroller.

Ekho substantially reduces the overhead of handling sensor data by a minimalist approach that uses a FIFO buffer between the sensors with RF analog front end. The FIFO buffer is the minimum element that is needed between sensing subsystem and communication subsystem to deal with

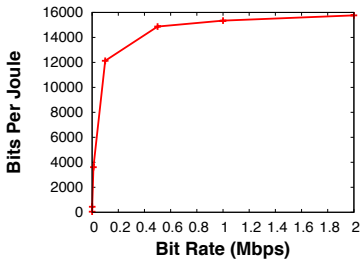


Figure 8: Efficiency of backscatter radio (in bits/joule).

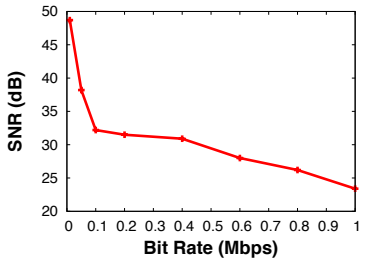


Figure 9: SNR at different bit rates when device is placed 1m from a reader.

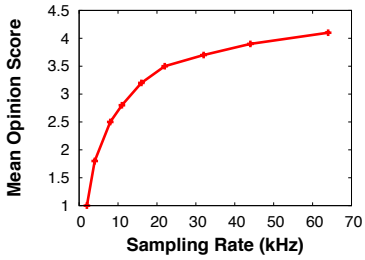


Figure 10: Mean Opinion Score (MOS) at different sampling rates for a microphone.

short delays in transmitting the data over the backscatter link, for example, due to intermittent scheduling of a device. In this manner, Ekho eliminates software and tail energy overhead that was observed on existing backscatter-based platforms.

The final computational component of the pipeline is the communication subsystem. Unlike EPC Gen 2 that is designed for a broad range of RFID tags, Ekho is designed solely for streaming sensor data from nodes to a reader. A protocol designed solely for streaming data from sensors can be quite simple. The reader informs each node of a timer value that specifies the period with which to transfer data in its FIFO buffer, and a rate that determines how fast to transfer the data. The only hardware component required for this protocol to work is a timer and shift register. Once the timer fires, a shift register converts the input sensor data to an ASK signal that is used to modulate backscatter radios.

In the current instantiation of Ekho, we do not perform any encoding of data. While the need for encoding to deal with harsh wireless conditions and interference is well-known, it also makes the hardware more complex, and consequently more power hungry. For example, the default configuration on the UMass Moo/UW WISP platforms is Miller-4 encoding incurs overhead of several hundreds of gates. Thus, while encoding may be useful in some cases, we do not employ it in Ekho.

4.2 The EkhoNet MAC layer

We now turn to the second part of our performance puzzle — achieving high throughputs that are upwards of many hundreds of kilobits/second across different nodes in the network. A high speed MAC is important for supporting an architecture where raw data transfer is the norm rather than the exception.

MAC layer designs are very well understood, particularly in cases such as ours where a central controller performs TDMA-like scheduling of sensor nodes. However, the key point in our design is two-fold: a) even though the sensor node is designed to be extremely simple, the decision making logic can be placed at the reader, thereby enabling surprisingly complex scheduling mechanisms across a network of extremely simple sensor nodes, and b) our MAC is holistic in that it takes into account utility of data, channel-awareness, energy consumption, as well as other hardware considerations, in-order to maximize throughput.

4.2.1 MAC Design Considerations

At the heart of EkhoNet is the logic that is used to determine when each node should transfer, and what rate they should transfer. Before we answer this question, we need to understand several characteristics of Ekho including: a) how do MAC-layer parameters impact the energy-efficiency of the platform? b) what are the signal-to-noise ratios at which data transmitted by Ekho can be successfully decoded? c) what criteria should we use to decide what sampling rate to use when sufficient bandwidth is not available? and d) what are the implications of platform considerations such as clock drift and buffer size? We now empirical examine these considerations in greater detail, and discuss the implications on selection of MAC layer parameters.

Bits/Joule: The first question we ask is how energy-efficiency of data transfer depends on the bit rate. Figure 8 shows the efficiency of a shift register controlled backscatter radio across different bit rates. At low rates, there is a steep increase in efficiency as bit rate increases due to the fact that constant power consumption by the system is amortized over more bits being transferred. However, improvements in efficiency diminish once the bit rate increases beyond 1Mbps since the relationship between power and frequency of the shift register is roughly linear, hence there are not much improvements possible. The power curve suggests that, from energy perspective, we should choose the fastest bit rate possible for data transmission.

Signal to Noise Ratio: While faster bit rates are preferable due to higher energy efficiency of transfer, SNR degrades as bit rate increases. Figure 9 shows the SNR when we deploy a transmitter 1 meter from the reader and change its transmission bit rate. As bit rate increases, the SNR decreases steadily as one would expect. When the SNR is lower than 10dB, decoding becomes difficult on our software-defined radio based reader platform, which gives us an upper bound on the fastest bit rate that can be supported by the system without losing bits.

Utility of data: Since EkhoNet is designed for high-rate sensors, one question that needs to be addressed is how to decide on appropriate sampling rates when the overall data rates at full sampling rates exceed capacity. On our existing system, we are limited to 1Mbps aggregate transfer rate across all nodes since the SDR-based reader is only able to support 8M samples per second due to the limitations of the realtime signal processing logic. This means that we can easily reach the SDR limit when we operate a network of sensors. For example, a network of five audio sensors sampling at 44 KHz, and transmitting raw data generates an

aggregate bandwidth of 3.5Mbps, well above what can be supported by EkhoNet.

Our solution is to take into consideration the utility of data generated by the different sensor nodes. Figure 10 shows an example of one utility function, Mean Opinion Score (MOS), which is a commonly used metric for characterizing the quality of transmitted audio [5]. The MOS score can be used to guide decisions regarding which node is allocated bandwidth.

Clock drift: Another consideration in determining slot sizes is clock drift. For example, in our implementation of Ekho, we use a crystal oscillator driven system clock that can drift at upwards of 50 ppm. If two nodes transferred at 1 Mbps, then they would drift by 1200 clock cycles each minute. The reader can handle clock drift in two ways. First, when assigning slots, it can allocate guard bands in each slot to allow for some drift. However, guard bands should be kept to a minimum to reduce bandwidth wastage. Second, the reader has the luxury of observing how the gap between slots varies as nodes transfer, and can detect when collision occurs by looking at the constellation plot of the signal [22]. Thus, when the reader suspects that slots have bled into each other, it can send a reset pulse that informs all nodes to reset their timers. Note that this is possible for backscatter because reader messages are broadcast and received by all nodes. A reset pulse is simply implemented by shutting off the carrier for a short, pre-defined duration, which is detected by each node. While reset pulses can be short, it should be used infrequently since there can be robustness issues if a node does not receive the pulse. This can result in further collisions resulting in more reset pulses until the network synchronizes.

Buffer size: One additional constraint introduced by the Ekho hardware platform is that the FIFO buffer size on the device is limited, hence if the slot sizes are too long, samples will be lost since the buffer will overflow.

4.2.2 Channel-Utility-Energy aware Rate Selection

Given the above constraints, the overall problem that the reader faces can be described as follows: select the optimal bit rate and slot size such that aggregate utility of received data is maximized and aggregate energy consumption minimized, subject to constraints on the buffer sizes, SNR, and guard bands. We formalize this problem below.

We assume that the following parameters are given:

- The minimum SNR, 10 dB in our system, at which the reader can decode bits with low bit-error rate.
- The maximum achievable bit-rate r_i that is higher than the minimum SNR.
- The maximum sampling rate of each node $s_{max}(i)$.
- The size of each sample in bits, b, bits/sample.
- The fraction of each slot that should be a guard band δ.

Given these values, we need to choose the sampling rates for each sensor s_i, and the fraction of time allocated to each node t_i by taking into account the following objective:

- $\sum_{i=1}^{n} U(\mathbf{s})$ which is a measure of the aggregate utility obtained from all sensor data received from the nodes.

The constraints are the following:

- $\sum_i t_i \leq 1$, i.e the fraction of time allotted to nodes sum up to at most one (less than one if the network is operating below its limit).
- $s_i \leq s_{max}(i)$, which restricts the sampling rate for a sensor to be below the maximum.
- $(1-\delta)t_i r_i = b\, s_i$, which ensures that the production of data from the sensor, and transmission of data from the radio are matched i.e. the node can transmit what is being sensed. The term $(1-\delta)$ is present since there's a guard band for each slot.

The overall optimization is shown below (in vector form for compactness). Here, $\mathbf{s}$ and $\mathbf{t}$ are the vectors of sampling rates and the fraction of time allocated to each node, which need to be determined, and $\mathbf{r}$ is the vector of bit rates chosen for each node based on SNR. The symbol $\preceq$ stands for element-wise inequality (i.e. one for each node).

$$\begin{aligned} \underset{\mathbf{s},\mathbf{t}}{\text{maximize}} \quad & \mathbf{1^T} U(\mathbf{s}) \\ \text{subject to} \quad & \mathbf{t^T 1} \leq 1 \\ & \mathbf{s} \preceq s_{max}\mathbf{1} \\ & (1-\delta)\text{diag}(\mathbf{t})\mathbf{r} = b\,\mathbf{s} \end{aligned}$$

Typically, the utility function is concave, for example in the case of MOS score (Figure 10). Hence, the objective is to maximize the sum of concave utility functions, and the constraints are linear, hence the optimization can be solved by standard convex optimization methods. Note that the optimization returns the fraction of time for each node — this can be converted to an actual slot size by scaling by an appropriate period such that each node is capable of buffering the data in its local FIFO buffer.

5. IMPLEMENTATION

Figure 11 shows the prototype of Ekho, which implements all the design elements described in section 4. The current prototype measures 1.8 by 2.4 inches, but we believe future revisions can shrink this even further. We now briefly describe the key sub-components used in the prototype.

5.1 Hardware

The first key hardware element is an ultra low power FPGA (Igloo Nano AGLN250) that manages the various sub-components of the Ekho platform. Most key components of the Ekho architecture, including the sensing, data handling, and communication subsystems, are implemented within the FPGA. The particular FPGA was chosen because it has low static current consumption and has a 32k bits (2KB) RAM, which also determines the maximum size of our FIFO buffer.

The next key design element is the backscatter circuit that can operate at high speed. As the device toggles the state of a transistor that connects to the antenna, an OOK signal that carries modulated information is generated. However, on existing backscatter platforms, the static current of the transistor is provided by the harvested RF energy, which might vary across time. The varying RF power affects the amount current that is provided to the transistor and leads to unstable edges of the generated OOK signal. Therefore, decoding becomes challenging when the data rate is high.

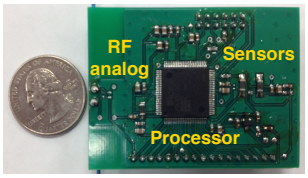


Figure 11: Ekho is implemented as a low-profile printed circuit board with small form factor.

Our backscatter circuit directly provide a small bias current to the transistor and retains a sharp edge for the generated OOK signal.

A critical element of our hardware design is the clock system which drives the FPGA logic. The core of our clock system is a 1MHz ultra low power crystal oscillator that directly feeds into the FPGA. The 1MHz clock is divided to drive different components of the architecture because sensing, data handling, and communication subsystems operate at different speeds. Our clock system is different from the Moo and WISP platforms, where a digital generated clock (DCO) is used. Although the DCO can also be divided for driving different components, it couples the operational modes of the system and its clock speed, as a result of which the high speed clock is only available when the system operates as a whole in a high power mode.

5.2 Software defined backscatter reader

We used the USRP N210 mother board and the SBX RF daughterboard to build our software defined backscatter reader for receiving high speed backscatter signals from Ekho. We construct a signal processing pipeline that is able to track the amplitude of the carrier wave that is used as the reference for decoding the OOK signal generated by Ekho. Our decoding is different from Moo and WISP platforms where Miller-4 encoding is used on top of the OOK signal and a decoding template can be used for correlating the received signal and output a bit when the template matches the received signal. In Ekho, the data is sent directly via OOK and encoding is not used. Therefore, we need to track the amplitude of carrier wave to determine whether the received signal is a high or low pulse.

5.3 MAC layer protocol

Figure 12 shows the timing diagram of the Ekho MAC layer. The first stage is to inventory the nodes in the network, and obtain information about their SNR and other sensor-related information. This phase executes very similar to an EPC Gen 2 singulation phase, where nodes can select a slot to transfer in, and send a short sequence of bits with the appropriate information. After the singulation phase, the reader executes the optimization algorithm described in §4 and determines the time period and bit rate for each sensor, which is then relayed to the sensor. The reader initiates the singulation phase under several circumstances: a) when significant changes are observed in SNR, which might signify changes in position or orientation, and b) when collisions are detected, which might signify that a new node is attempting to join the network.

Once the reader informs each sensor of its bit rate and period, it initializes slots by sending a synchronization signal during which it shuts down the carrier for a short 10 μs

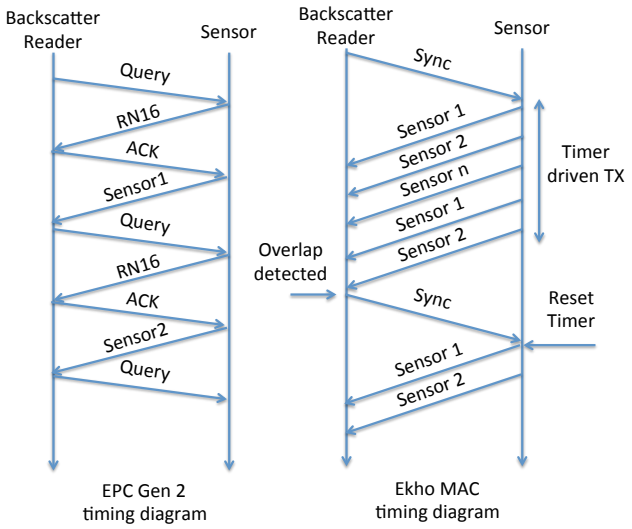


Figure 12: Timeline of Ekho MAC.

window. This pulse informs all nodes simultaneously that they should start their timers, thereby initiating the TDMA schedule. The length of the sync message needs to be chosen small enough to amortize overhead, but large enough to be detectable at the sensor, hence our choice of 10μs.

When the reader detects that data transmitted during adjacent slots are overlapping into each other (due to clock drift), it re-issues a synchronization pulse to restart the timers on all nodes. Overlap between sensors can be detected by looking at the constellation map of the received signal — if two clusters are present, it indicates that a collision-free signal is received and if more clusters are present, it indicates that a collided signal is observed [22]. If multiple synchronization pulses fail to eliminate collisions, the reader switches back into inventory mode.

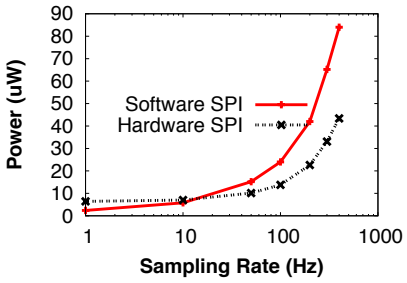
6. EVALUATION

We now evaluate the overall performance of EkhoNet including 1) demonstrating the power benefit of the Ekho architecture, 2) benchmarking the performance of the EkhoNet MAC, and 3) evaluating EkhoNet's ability to support high-rate streams from many sensors while operating at extremely low power consumption.

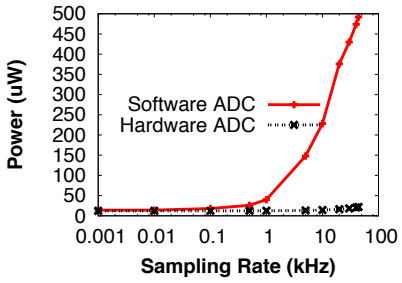
6.1 Experimental setup

We deploy 10 Ekho nodes 1 feet to 9 feet from a backscatter reader. Our experiments do not cover distances larger than 9 feet because of the poor signal quality beyond 9 feet. This is a result of the 100mW maximum power issued by the SBX RF daughterboard, which is 10× smaller than commercial RFID readers.

To understand the power benefits of Ekho, we compare against the UMass Moo (equivalent of Intel WISP 4.0) and the WISP5.0 platforms. Since the WISP5.0 platform is not currently available, we evaluate its power consumption with a prototype that uses the same MCU (MSP430FR5969). Since the MCU is the main power hog in the system, this provides a good proxy for measuring power consumption.



(a) Accelerometer.



(b) Microphone.

Figure 13: Power reduction for sensing subsystem: a) sampling an accelerometer, b) sampling a microphone.

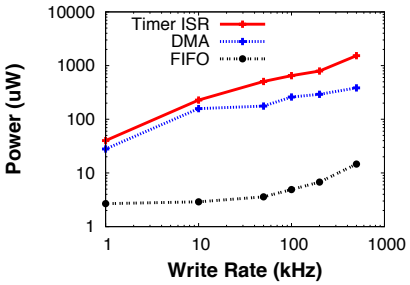


Figure 14: Power reduction for data transfer to network queue.

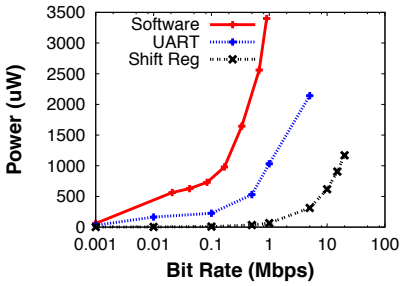


Figure 15: The power consumption of operating a backscatter radio.

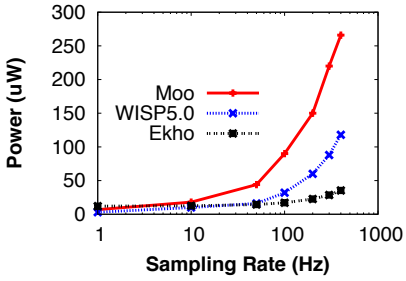


Figure 16: Whole-system power consumption for operating an accelerometer sensor.

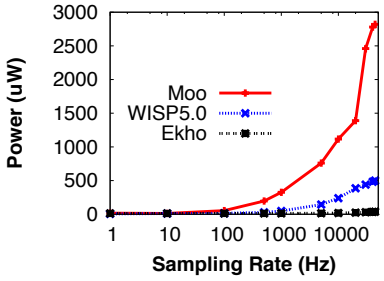


Figure 17: Whole-system power consumption for operating an audio sensor.

6.2 Ekho power benchmarks

We begin our evaluation by validating the claim that the power optimizations on Ekho can substantially reduce the overheads incurred by existing platforms. We follow the organization in §4, and show benchmarks for each module — sensor data acquisition, sensor data handling, and network stack.

Figure 13 measure the power of the sensing subsystem when Ekho interacts with two types of sensors — an accelerometer with on-board ADC that connects to the MCU via a SPI interface, and an audio sensor where the MCU's ADC is used to sample the sensor. We compare Ekho versus a WISP/Mote-class sensor device (i.e. a device where the sensor connects to an MCU that acquires data). In both cases, we can see that Ekho reduces power consumption substantially — for sampling the accelerometer, Ekho reduces power by 1.5× at 400Hz by eliminating the overhead of software-controlled SPI, and for sampling the audio sensor, Ekho reduces power by 22× by trimming the overhead of running the software-controlled ADC at 44kHz.

Figure 14 measures the power consumption of the data handling subsystem of Ekho, which is composed by a 2kB FIFO buffer for connecting sensors to the RF analog front end. The 2kB FIFO buffer only consumes 26.5μW of power when data is written into the FIFO at 500kHz, 14.4× lower than the 384μW consumed by DMA driven data migration and 92× lower than the 1.5mW consumed by timer driven data migration.

Figure 15 shows the power consumption of the communication subsystem which is composed of a shift register and backscatter radio. At 1Mbps, Ekho's communication subsystem consumes only 77μW of power, 13.4× lower than a UART controlled backscatter radio implemented on the WISP and 44× lower than a software controlled backscatter radio implemented on the WISP. For software and UART controlled backscatter radios, we do not measure power at bit rates higher than 6Mbps because the maximum clock on rate on WISP platform is 24MHz, which limits the maximum achievable bit rate.

6.3 Whole-system power consumption

Having looked at power benchmarks for individual components of Ekho, we turn to a whole-system power measurement from sensing to transmission. We look at the overall power consumed by Ekho when operating the same two sensors as earlier — accelerometer and microphone.

We start with a measurement of Ekho with an accelerometer. The sensor has a built-in ADC and talks via SPI to the sensor platform. Figure 16 shows that at 1Hz, the power consumption of Ekho is higher than Moo and WISP5.0 platforms. This is because the static current consumption of the FPGA at the core of Ekho is 8.9μA, much higher than the 0.1μA static current draw of Moo and WISP5.0. However, when the frequency of operating the accelerometer increases, the power consumed by Moo and WISP5.0 platforms increases significantly while the Ekho system still consumes only tens of μW. At 400Hz, the Ekho system consumes 35μW of power, 7.6× lower than the 266μW of Moo and 3.3× lower than the 118μW of WISP5.0.

We now turn to power measurements when Ekho is connected to a microphone. An external ADC is used to sample the audio sensor, and send a digital signal to the core plat-

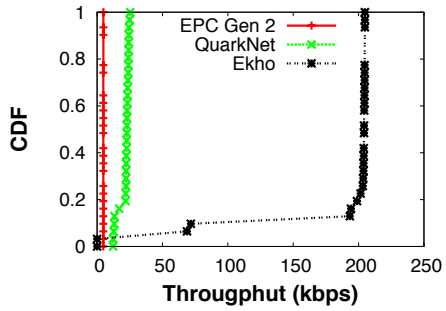


Figure 18: Comparing throughputs of EPC Gen 2, QuarkNet on Moo vs Ekho across 30 locations.

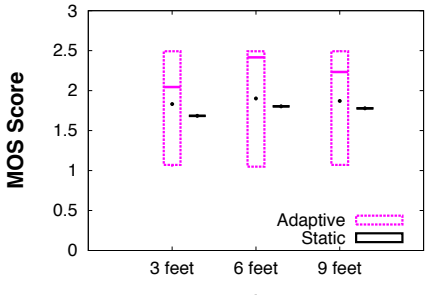


Figure 19: Boxplot of the MOS scores for 10 Ekho nodes with microphones at 3 locations (3 ft, 6 ft, 9 ft).

form (Moo, WISP5.0, or Ekho). Figure 17 shows the power consumption of the three platforms. At 44kHz, the Ekho system only consumes 37μW of power, 76× lower than the Moo and 13.5× lower than the WISP5.0.

In conclusion, Ekho is particularly efficient when using higher rate sensors that sample at frequencies of hundreds of Hz. A crucial observation is that even when the sensing rate increases by two orders of magnitude from the accelerometer at 400Hz to the microphone at 44kHz, the overall power consumption remains almost the same. This shows that Ekho scales up very well as sampling rate increases. In addition, Ekho is able to operate with sensors that use SPI or provide an analog signal while retaining high efficiency.

6.4 Evaluating EkhoNet's throughput

Having discussed the power benefits of Ekho, we now turn to look at the performance of Ekho's transfer rate.

We start with the throughput achieved by a single node. Since the Moo and WISP platforms currently support only a 256Kbps baud rate, we fix Ekho's clock to operate at the same rate. We then compare Ekho's throughput against the Moo executing EPC Gen 2 [24], and QuarkNet [25]. Figure 18 shows the cumulative throughput across 30 locations. The 30 locations are chosen randomly between 1 feet to 9 feet from a backscatter reader.

There are two key observations. First, we see that the throughput achieved by Ekho is 45× higher than Gen 2 and 8× higher than QuarkNet on the Moo. EPC Gen 2 suffers greatly due to protocol overhead, and therefore achieves abysmal overall throughput. Although QuarkNet is a highly optimized system that is designed for micro powered sensors, its throughput is limited by the fact that the PHY layer (encoding, etc) is implemented in software on Moo, which reduces throughput. Second, we see that there are a few locations where our design decision to eschew encoding hurts us. At those locations, the received signal can still be decoded by EPC Gen 2 and QuarkNet because of the SNR benefit of Miller-4 encoding. However, it can be seen that this is a small fraction of the overall range of the reader. (Note that if encoding is essential, it is possible to add this module to Ekho at the cost of some additional power consumption and reduced throughput.)

We now turn to the throughput achieved by a network of nodes, and evaluate the benefits of our energy and utility function aware bit rate selection algorithm. We deploy 10 Ekho nodes with microphones at three locations (3 feet, 6 feet, and 9 feet from a backscatter reader). The maximum sampling rate of each audio sensor is 44kHz and each sample data is 16 bits. As a result, an audio sensor can generate up to 706k bits data per second. In contrast, the overall network transmission capacity of EkhoNet is 1Mbps in our current instantiation since each device is equipped with a 1Mbps clock. Thus, 10 audio sensors in front a backscatter reader can saturate the 1Mbps network easily, which means that adapting the bit rate as well as the sampling rate of each sensor is necessary.

When channel is saturated, the selection of bit rate is intuitive because maximum bit rate which meets the lowest SNR decoding threshold (10dB) should be used. The selection of sampling rate follows the energy-utility joint optimization we formulated in §4.

Figure 19 shows the MOS score obtained by 10 audio sensors at 3 locations. Our optimization framework attempts to allocate bandwidth such that sensors with higher SNR can get the bandwidth they need for achieving higher MOS scores. As a baseline, we compare against a scheme that allocates bandwidth equally across all sensors. The median and mean MOS scores achieved by EkhoNet is higher than the baseline scheme — 50% of the nodes have MOS scores higher than two, which is acceptable audio quality, whereas the uniform allocation scheme has MOS scores of about 1.7, which means poor audio quality. A breakdown across nodes shows that our algorithm assigns higher sampling rate to sensor 1 to 5 because they have higher SNR. While other application-specific utility functions are possible, these results demonstrate that despite the simplicity of Ekho platforms, the EkhoNet MAC can be more complex and optimize network-wide throughput, energy and utility.

7. RELATED WORK

Backscatter communication: There has been much recent emphasis on backscatter communication. Some efforts have explored bandwidth limitations of backscatter communication in terms of throughput including Flit [13], Buzz [22], and Blink [27]. While there are interesting ideas underlying each of these, the overall throughput achieved by EkhoNet is orders of magnitude higher than the above systems as a result of a clean-slate design. Other efforts have focused on using harvested power in an efficient manner including QuarkNet [25][26] and Dewdrop [8] — these approaches are complementary to EkhoNet and can be used in conjunction with the ideas in this paper.

In addition to the above, there have been many interesting ideas on using backscatter for real-world applications. Ambient Backscatter [20] uses the backscatter of FM signals for short-range communication between tags to enable credit-card transactions. AllSee [18] explores the backscattered signal for gesture recognition. These ideas can potentially benefit from an Ekho-like platform that is designed to reduce power consumption while increasing bandwidth.

Much literature has explored the design of MAC layer protocols for RFIDs, and several of these approaches specifically address data collection from RFID-scale sensors [9, 13, 22, 25]. Viewed in isolation, our MAC layer protocol is simplistic since its merely a stripped down version of TDMA, hence it relates to most of the above protocols. However, our work should be viewed not just as a MAC layer, but a system-wide re-design to strip computational overhead from backscatter-based sensors, and thereby achieve higher efficiency.

Optimized sensing platforms: There have been many highly optimized sensor hardware designs proposed over the past decade. At a high level, these can be separated into two classes — optimized hardware platforms designed for specific applications, and optimized hardware platforms that are intended as a building block for research and applications. One example in the former class is the NeuralWISP [16], a wireless neural interface that operates on harvested RF energy. Some examples in the latter class are the Michigan M^3 [19], an impressive mm^3 sensor that operates at low power, and the Epic Mote [11], which is a modular mote-class platform for enabling low-power wireless sensor network applications.

EkhoNet differs from these efforts in that it is designed for raw data transfer from high-rate sensors at extremely low power levels. Thus, it is a general-purpose platform for sensors similar to the second class of devices, but focused on backscatter and high-rate sensors. As a result, the underlying design principles and optimizations are completely different from those that drive the other class of platforms.

8. DISCUSSION

While Ekho provides substantial performance benefits over the state-of-art in backscatter-based sensor platforms, there are several questions that we have not completely addressed in our evaluation. We discuss these in this section.

FPGA v.s. MCU: One of the design choices in Ekho is the use of an FPGA rather than MCU — this choice greatly reduces the computational and data migration overheads between the sensor and radio, but in the process, it sacrifices ease of programmability. While FPGA programming has become easier in recent years due to improved IDEs and GUI interfaces [4], it requires familiarity with logic design at the circuits level. MCUs, on the other hand, are much more natural to program using commonly used high-level languages such as C, which is one of the reasons for its wide use on sensor platforms.

We believe that the greater difficulty in programming FPGAs is not as much of an issue for Ekho as for other platforms. Wireless sensors are designed to be intelligent, autonomous nodes that can adapt to dynamics in energy levels, channel conditions, routing changes, and others. In contrast, Ekho is designed to be a "dumb" peripheral for a powerful reader that simply forwards the raw sensor data over a backscatter link. Much of the decision-making logic that is traditionally implemented on the sensor side are performed at the reader. Thus, Ekho can be viewed as just another sensor, with an interface that allows the reader to set sampling rates and bit rates (as shown in §4).

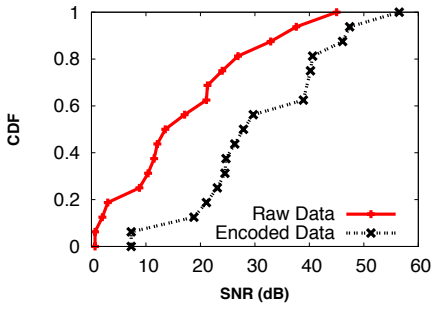


Figure 20: SNR of transmitting encoded data and raw data across 20 locations.

Power benefits: The results presented in this paper compare Ekho against existing backscatter-based sensing platforms such as the WISP, but one question is whether we would have significant power benefits if we compared against an FPGA implementation of the WISP. Our evaluation did not address this question since re-implementing the entire sensing, computation, and communication pipeline of the WISP on an FPGA is a substantial effort, but we provide a qualitative comparison.

Existing research work [21] on RFIDs suggests that an EPC Gen 2 tag implemented on FPGA usually consumes 5K to 10K logic gates. Clearly, an EPC Gen 2 tag does not perform any operation related to sensing. Therefore, sensor sampling, data migration, buffering, and other tasks would incur additional overhead. For example, Touhafi and Glesner et al [10, 15] investigate an FPGA (Spartan3-2000) based sensing platform which consumes 1200K gates, several orders of magnitude higher than an EPC Gen 2 tag. Our Ekho implementation consumes only 6K gates, which is comparable to an EPC Gen 2 tag and significantly less than what we would expect with an FPGA version of the WISP. Since the power consumption of an FPGA depends on the number of gates used, Ekho should still be significantly more efficient.

Encoding: Another design decision that needs more discussion is that Ekho eschews encoding in an effort to be minimalist. Unsurprisingly, this can be problematic in scenarios where the wireless channel is noisy. Figure 20 shows a simple experiment where we place a tag at 20 locations between 1–9 ft in front of a reader, and look at the SNR with EPC Gen 2's Miller-4 encoding, and without encoding. The decoding threshold for our backscatter reader is 10dBm, so any signal lower than this threshold cannot be decoded correctly. As expected, there is about a 10dB difference between encoded and uncoded signal. The SNR is higher than 10dB in 80% of the locations for uncoded data, and higher than 10dB in about 90% of the locations after encoding. This comes at a high cost, however, since the node consumes 8× more power for achieving the same bit rate.

Thus, our point is simply that encoding is yet another computation block on a backscatter-based sensor platform. While the power consumption of techniques like encoding

are insignificant in most radios, the pros and cons deserve to be examined more carefully for ultra-low power platforms such as Ekho.

Applications: Finally, this paper does not focus on applications of Ekho, but we view our work as an enabler for a variety of applications. While the idea of backscatter-based sensing is not new [23], many existing efforts are about networking simple, low rate sensors (e.g. temperature, pressure, etc). But the need for backscatter in such scenarios is debatable — active-radio based wireless sensors operate for years on coin cells at low sensing and communication rates. But rich sensors such as microphones and cameras operate primarily in a tethered manner since data rates are far too high for continuous communication. Our work seeks to bridge the gap, and enable camera networks or microphone networks to stream data continuously in an untethered manner. The benefits of streaming raw sensor data to internet-connected infrastructure is immense since one can use vast amount of computational resources to jointly process the data streams and enable smart applications. A simple example would be continuous speaker recognition and transcription of meeting notes by deploying a tethered reader and dozens of untethered Ekho nodes at different locations in a conference room.

9. CONCLUSION

In this paper, we present a powerful backscatter wireless sensing architecture, Ekho, that can sample sensors at tens of kHz and transmit data wirelessly at several hundreds of kbps, while only consuming tens of μWatts of power. The key observation in Ekho is that backscatter wireless communication is energy-wise much cheaper than computation. Therefore, by eliminating the overheads of sensing subsystem, data handling subsystem, and communication subsystems, we enable the whole sensing-communication pipeline to operate at extremely low power. Over the Ekho platform, we design a MAC layer that allocates bit-rates across nodes while taking into account energy-efficiency, utility of data, and a variety of platform-level considerations. We believe that EkhoNet can enable new explorations in backscatter-based sensing systems, and enable new applications that use ultra-low power high-rate sensors.

Acknowledgements

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

10. REFERENCES

[1] Analog devices adxl362 accelerometer sensor. `http://www.analog.com/static/imported-files/data_sheets/ADXL362.pdf`.

[2] Analog devices mems microphone admp803. `http://www.analog.com/static/imported-files/data_sheets/ADMP803.pdf`.

[3] The cbvs ecg-on-chip. `http://www.clearbridgevitalsigns.com/chip.html`.

[4] Libero ide. `http://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-ide`.

[5] Mean opinion score. `http://en.wikipedia.org/wiki/Mean_opinion_score`.

[6] WISP: Wireless Identification and Sensing Platform. `http://seattle.intel-research.net/wisp/`.

[7] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *ACM SIGCOMM IMC 2009*.

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

[9] M. Buettner and D. Wetherall. A software radio-based uhf rfid reader for phy/mac experimentation. In *RFID (RFID), 2011 IEEE International Conference on*, pages 134–141. IEEE, 2011.

[10] A. De la Piedra, A. Braeken, and A. Touhafi. Sensor systems based on fpgas and their applications: a survey. *Sensors*, 12(9):12235–12264, 2012.

[11] P. Dutta and D. Culler. Epic: An open mote platform for application-driven design. In *IEEE IPSN 2008*.

[12] J. Gummeson, S. S. Clark, K. Fu, and D. Ganesan. On the limits of effective hybrid micro-energy harvesting on mobile crfid sensors. In *ACM MobiSys 2010*.

[13] J. Gummeson, P. Zhang, and D. Ganesan. Flit: a bulk transmission protocol for rfid-scale sensors. In *ACM MobiSys 2012*.

[14] S. Hanson, Z. Foo, D. Blaauw, and D. Sylvester. A 0.5 v sub-microwatt cmos image sensor with pulse-width modulation read-out. In *JSSCC 2010*.

[15] H. Hinkelmann, A. Reinhardt, S. Varyani, and M. Glesner. A reconfigurable prototyping platform for smart sensor networks. In *Programmable Logic, 2008 4th Southern Conference on*, pages 125–130. IEEE, 2008.

[16] J. Holleman, D. Yeager, R. Prasad, J. R. Smith, and B. Otis. Neuralwisp: An energy-harvesting wireless neural interface with 1-m range. In *BioCAS'08*, pages 37–40. IEEE, 2008.

[17] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *ACM MobiSys 2012*.

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

[19] Y. Lee, G. Kim, S. Bang, Y. Kim, I. Lee, P. Dutta, D. Sylvester, and D. Blaauw. A modular $1mm^3$ die-stacked sensing platform with optical communication and multi-modal energy harvesting. In *ISSCC 2012*.

[20] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith. Ambient backscatter: wireless communication out of thin air. In *ACM SIGCOMM 2013*.

[21] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda. LamedÑa prng for epc class-1 generation-2 rfid specification. *Computer Standards & Interfaces*, 31(1):88–97, 2009.

[22] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk. Efficient and reliable low-power backscatter networks. In *ACM SIGCOMM 2012*.

[23] D. J. Yeager, P. S. Powledge, R. Prasad, D. Wetherall, and J. R. Smith. Wirelessly-charged uhf tags for sensor data collection. In *RFID, 2008 IEEE International Conference on*, pages 320–327. IEEE, 2008.

[24] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational rfid and sensing platform. Technical report, Tech. Rep. UM-CS-2011-020, 2011.

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

[26] P. Zhang, D. Ganesan, and B. Lu. Quarkos: Pushing the operating limits of micro-powered sensors. In *Proceedings of the 14th USENIX conference on Hot Topics in Operating Systems*, pages 7–7. USENIX Association, 2013.

[27] P. Zhang, J. Gummeson, and D. Ganesan. Blink: a high throughput link layer for backscatter communication. In *ACM MobiSys 2012*.