QuarkOS: Pushing the operating limits of micro-powered sensors

Pengyu Zhang, Deepak Ganesan, Boyan Lu
{*pyzhang, dganesan*}*@cs.umass.edu, blu@engin.umass.edu*
University of Massachusetts Amherst

Abstract

As sensors penetrate into deeply embedded settings such as implantables, wearables, and textiles, they present new challenges due to their tiny energy buffers and extremely low harvesting conditions under which they need to operate. However, existing low-power operating systems are not designed with the goal of scaling down to such severely constrained environments. We address these challenges with QuarkOS, an OS that scales down by carefully fragmenting every communication, sensing, and computation task into tiny quanta (e.g. half-bit, one pixel) and introduces sleeps between such quanta to recharge. In addition QuarkOS is designed to have minimal run-time overhead, while still adapting performance to harvesting conditions. Our results are promising and show continuous communication from an RF-powered CRFID can occur at a third of the harvesting levels of prior approaches, and for image sensing to be performed with a tiny solar panel under natural indoor light.

1 Introduction

Technology trends in self-powered sensors are enabling them to be deployed in more deeply embedded settings than before, including miniaturized wearable and implantable bio-sensors that continuously measure physical and physiological parameters [14] [15], perpetual deployments of sensors embedded within buildings and pipes [11], miniature robotic insect swarms [6], and others. In addition, emerging 3D nano-fabrication techniques [5] promise cheaper sensors that can be printed in a cost-effective manner at high volumes, enabling applications such as smart textiles and smart surfaces. In most of these scenarios, large energy buffers are infeasible due to form-factor and cost considerations, and are instead replaced by a tiny rechargeable energy storage and a micro-harvester that scavenges energy from light, electro-magnetic, vibrations, and/or temperature [3]. By combining such tiny harvesters with low-power electronics and manufacturing, the aim is to achieve affordable, perpetual, and miniature form-factor sensor systems.

System design for micro-power based sensors is extraordinarily challenging for two reasons — small buffer size and tiny harvesting rate. The small form-factor demands of deeply embedded sensors mean that micro-powered sensors rely more on continuous harvesting and less on energy storage. In turn, this places substantial burden on a programmer to carefully construct tasks such that they can be executed within the buffer constraints. The second challenge is that micro-harvesters provides energy in tiny trickles and in a time-varying manner. This results in several idiosyncrasies that the system needs to handle: a) harvesting rate is often considerably lower than the active mode power draw, hence outages can occur, b) regulation and voltage boosting is expensive, hence harvesting systems need to adapt to unregulated power from the harvester, and c) even simple operations such as ADC reads to measure energy in the buffer are expensive at the micro-scale, and need to be avoided.

In this paper, we look at design abstractions that can enable an operating system for micro-powered systems to scale down to the most extreme harvesting conditions and the most stringent buffer size limitations. Traditional sensor operating systems such as TinyOS [9] and Contiki [7] are designed for battery-powered systems, and use larger tasks that simply do not scale down to these regimes. More recently, MementOS [13] and Dewdrop [4] address some of the challenges in micro-powered environments. MementOS introduces checkpoints within computation tasks such that it can recover from outage and continue execution. Dewdrop is a system that adapts task execution to harvesting conditions such that the efficiency of execution is optimized. While these are important steps to deal with the constraints in micro-powered systems, they do not scale down with the buffer size and available energy. For example, both of these systems execute on a light-powered Intel WISP placed in the presence of strong light (2000 lux), but placing it under natural indoor light that is a factor of ten smaller (200 lux) makes these systems inoperable.

In this paper, we argue for a simple but powerful abstraction — we deconstruct every task within the operating system and look for opportunities to insert sleep gaps to allow for the buffer to replenish before continuing execution of the task. We look at packet transfer, and identify opportunities between bits as well as within a single bit where sleeps can be inserted. We look at complex sensors such as imagers, and identify how sleeps can be inserted between and within individual pixel sensing operations. Similarly, processing tasks can be broken down into smaller segments that may be only a few instructions in size. The ability to breakdown every possible task into the smallest contiguous execution unit allows QuarkOS to scale down to extremely power-deprived regimes where existing systems cannot operate.

Optimizing systems for resource-impoverished conditions can limit performance when scaling up to regimes where there is more available energy. This is because active to sleep transition overheads can dominate when each execution unit is extremely small. To avoid this, QuarkOS continually adapts to harvesting conditions, and adapts the size of each execution unit to ensure that transition overhead is minimized. Importantly, such adaptation is performed by avoiding power-hungry operations such as ADC reads, which can limit the conditions under which QuarkOS can operate.

Our preliminary results show that QuarkOS can scale-down to the most extreme of conditions: a) QuarkOS runs on a UMass Moo CRFID equipped with capacitors that are two orders of magnitude smaller than their default configuration, b) QuarkOS establishes continuous communication with a reader at a third of the harvested power that is required for existing CRFIDs, and c) QuarkOS enables continuous image capture while powered with a small solar harvester under natural indoor lighting conditions despite a limited buffer that only has enough charge to sample a single pixel.

2 Case for QuarkOS

At the heart of QuarkOS is a simple hypothesis — by breaking down every component of an OS (communication, sensing and computation) into its smallest indivisible units, we can enable the system to scale down to the most extreme resource-impoverished regimes and the most stringent platform constraints. We discuss some of the considerations that influence our design, as well as the limitations of prior work.

2.1 Why fragment tasks?

The need to break down every task in a system into its smallest units arises from two considerations — limitations of the platform and limitations of the harvesting source. The limitations enforced by the sensor platform are often driven by form-factor, cost, and fabrication capabilities. As sensors become more deeply embedded into bodies, buildings, and the environment, one of the sacrifices that platform designers are increasingly making is reducing the size of the energy buffer and substituting it with a micro-harvester. Table 1 shows that the buffer size shrinks by more than six orders of magnitude between a Mote sensor and its "smart dust"-sized counterpart, a Michigan Micro-Mote (M^3). Even across micro-harvesting platforms, there are differences — for example, the Intel WISP has a buffer that is an order of magnitude larger than the M^3 mote.

Table 1: Shrinking buffer size.

	Buffer size
Mica Mote [12]	2850mAh
CRFID (WISP/Moo)[2][16]	5.44uAh
Michigan Micro Mote (M^3)[10]	0.6uAh

The limitations of the harvesting source are across many fronts. It provides energy in small trickles that varies over time. For example, the M^3 mote is attached to a small single-cell solar panel that harvests about 10 uW of power and a CRFID (WISP) harvests about 400 uW at a distance of 1m from a reader. While trickle charging can be addressed by filling up the buffer and executing a task, it is often not possible to charge a buffer to its maximum capacity under extremely low harvesting conditions. To fully charge a buffer, the voltage/current has to be boosted up via regulators and boost converters, which are too power hungry to operate under such conditions. Therefore, many platforms (e.g. WISP, Moo) put the burden on the run-time system to adapt to varying levels of available energy.

The design of an OS for devices with tiny buffers and variable amounts of available energy is extraordinarily complex — even tasks as simple as transmitting a single packet may not be feasible on the energy available per charge cycle, rather communication may need to be broken down to a few bits or perhaps even a fraction of a bit at a time. Similarly, sensing tasks such as capturing an image or obtaining samples from an accelerometer may need to be broken down further in-order to operate.

2.2 Limitations of prior work

A wide range of operating systems have been proposed for Mote-class devices over the last decade including TinyOS [9], Contiki [7], Nano-RK [8], and others. These OSs are designed to operate on a finite buffer that can last several days, weeks, or months, and the core emphasis is on maximizing lifetime given such a buffer rather than optimizing the size of a task. For example, while programmers are encouraged to use small task sizes in TinyOS, this is only because there is no pre-emption across tasks, hence a large task can starve a smaller one which may have timing requirements. But while these tasks are relatively small compared to larger platforms, they are, in fact, orders of magnitude larger than what can be executed with buffers on micro-powered platforms.

More related to our regime is MementOS and Dewdrop, which have addressed challenges in RF-harvesting based Computational RFIDs (CRFIDs). MementOS [13] is an operating system for CRFIDs that introduces checkpoints within a task such that the it can continue execution after an outage. While MementOS fragments tasks across outages, it is limited in three ways: a) tasks that have timing requirements, for example communication and sensing, are difficult to split across outages due to the non-determinism of boot-up and shut-down operations, b) checkpointing using flash is expensive energy-wise compared to sleep, and c) flash requires relatively high voltage, and therefore this is not appropriate under extremely low harvesting conditions.

Dewdrop [4] is an energy-aware runtime system for CRFIDs where the system adapts to available energy and wakes up to run a task only when it believes that there is sufficient energy for the task to execute. While Dewdrop does not address task splitting, the harvesting-awareness capability is important to QuarkOS as well. However, Dewdrop uses ADC sampling for tracking harvesting rates, which is an energy-hog and renders the system inoperable under severely constrained settings.

2.3 QuarkOS Abstraction

QuarkOS addresses the above challenges by breaking down all tasks in the system into the smallest units that need to be contiguously executed — fraction of a bit, fraction of a sensing unit (e.g a part of a pixel), single byte of storage, and single instruction execution. This allows QuarkOS to operate under extreme scenarios — for example, on a device that only has enough energy to transmit one bit at a time, a packet can be transmitted over several tiny re-charge cycles of one bit duration. In between these execution units, QuarkOS introduces sleeps where the platform operates in sleep mode while it waits for the buffer to re-charge.

3 QuarkOS Design

In this section, we describe the key components of QuarkOS. We first describe how we can breakdown the communication and sensing stacks to operate under extremely constrained conditions, and then how the QuarkOS controller adapts to harvesting conditions in a lightweight manner.

3.1 Fragmenting the network stack

One of the key innovations in QuarkOS is the capability to break-down packet transmission such that it can operate one-bit at a time (more precisely, a fragment of a bit at a time). We focus on passive backscatter communication in this section since it is considerably lower power than active radios and therefore more appropriate for harvesting-based devices. We start by introducing relevant information about passive communication, and describe how it can be broken into one bit or half-bit quanta, and then discuss the challenges in designing a complete network stack over such an abstraction.

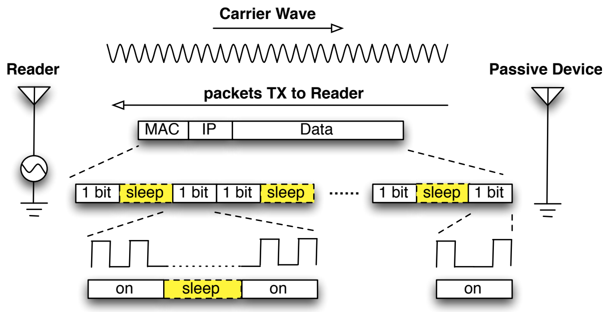

Figure 1: QuarkOS based bit-level networking.

Communication in passive radios A passive radio is designed to both provide power to a passive device as well as to enable communication. As shown in Figure 1, the reader provides a carrier wave, which can be reflected by a passive device back to the reader with its own information bits. The Intel WISP [2] and UMass Moo [16] are examples of sensor platforms that rely on passive radio. A unique difference between active and passive radios is that while active radios need an oscillator to clock the bits, the clock for passive communication is obtained from reader's carrier wave. The implication is that fragmenting passive communication into smaller quanta does not incur significant transition overheads since the oscillator is often the component that takes the most time/energy to stabilize.

Inter-bit and intra-bit sleeps QuarkOS exploits two opportunities to insert sleep gaps into a packet transmission (Figure 1): between bits and within bit. First, since hardware timers on the micro-controller are responsible for generating the OOK pulses on the passive radio, sleep gaps can be inserted between bits by clearing the hardware timers and putting micro-controller into low power mode. Second, sleep gaps can even be inserted within a single bit by setting the transmission unit as half bit. The conjunction of two half bits is carefully designed to maintain the timing information within each bit.

QuarkOS network stack While we have discussed the building blocks of a bit-by-bit network stack, several challenges remain. First, how can encoding and decoding be performed in a manner that is robust to sleep gaps? One approach that we are exploring is parallel decoding to compare each received analog signal against all templates (preamble, data 0 and 1) and distinguish data versus sleep gaps. Second, how can packet reception at a sensor be designed to be robust to sleep gaps and variability in the gap? One approach might be for the sensor to announce the wakeup duration, sleep gap, and jitter that it expects for the duration of the transfer, and for the transmitter to construct packets in a manner that is robust to these parameters. Third, the network stack will need to distinguish between energy-related losses and channel-related losses since different approaches need to be employed to recover from them. For example, at the MAC layer, energy-related losses would need to be handled by sleep duration adaptation whereas channel-related losses would need to be handled by bit-rate adaptation. Fourth, any part of the system that relies on timing information including CSMA channel idle time monitoring, and TTLs at the transport layer, will need to be designed to be aware of sleep durations. We are exploring these challenges in our ongoing work.

3.2 Fragmenting the sensing stack

Having described how communication tasks can be broken down into tiny fragments, we turn to sensing tasks. Micro-powered platforms are almost always designed with the ability to sense, hence the ability to breakdown complex sensing tasks is crucial to the design of QuarkOS. Our key insight is that opening up the inner workings of sensors allows it to be broken down into small sub-components with sleeps between them. We focus on one such sensor - an imager - because it represents a particularly complex sensor that would normally be considered far too power-hungry to be used on micro-powered platforms. The mechanism by which we break down image sensing parallels our approach for the network stack. We breakdown the individual components of an imager and look for opportunities to insert sleeps between small quanta such that the overall image sensing task can still be performed despite harvesting and buffer limitations.

Breaking down an imager A photo image sensor is a device that converts an optical image into electronic signals. To capture a whole optical image, the electronic signal at each pixel needs to be sampled. While most commercial imagers are designed to capture an entire image and transfer it to a processor, some recent low-power imagers allow greater control over pixel acquisition [1]. In addition, the sampling of a single pixel in such imagers can be broken down into several steps as shown in Figure 2: 1) configure corresponding sensor registers, 2) turn on the amplifier to increase the signal strength of generated electronic signal, 3) perform ADC readings to convert the electronic signal into digital data, and 4) turn off the amplifier and the ADC. The decomposed pixel sensing operations provide several potential opportunities to insert sleep gaps.

Inter-pixel and intra-pixel sleeps QuarkOS exploits two opportunities to insert sleep gaps into an image sensing: a) sleeps can be introduced between pixel acquisitions during which the power-consuming units such as amplifiers and ADCs can be switched off, and b) sleeps can be inserted at an intra-pixel granularity before turning on the amplifier or after the ADC readings to restore energy in the buffer. Thus, the minimum energy requirements for image sensing is only the amount needed to sense a fragment of a pixel at a time.

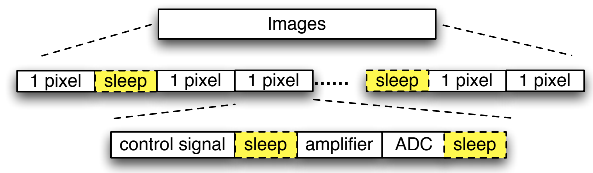

Figure 2: QuarkOS based pixel-level image sensing.

QuarkOS image processing stack While we have discussed the building blocks of a pixel-level image sensing stack, several challenges remain in designing a robust sensing stack over this abstraction. For example, when capturing a dynamic scene, the reduced sampling rate will result in motion blur. Handling motion blur is particularly complex if the sleep gaps vary over time due to harvesting source variations. One approach to address this problem might be to exploit a regression model to learn the time series patterns of inserted sleep gaps and compensate the corresponding blur according to the model prediction. Another might be to limit the scenarios under which images are captured to those where harvesting conditions are relatively stable.

3.3 QuarkOS interface and runtime

Fragmentation of tasks in QuarkOS has implications both on application design as well as the run-time system.

Application Interface An application developer can use QuarkOS to execute a variety of sensing, processing, and communication tasks while remaining agnostic of how the tasks are fragmented. However, QuarkOS can introduce variable-sized gaps depending on harvesting conditions, which impacts timing-sensitive tasks. For example, when capturing an image under dynamic conditions, sleep gaps result in motion blur and variable gaps can make the image difficult to interpret. To address this, tasks can specify the maximum sleep gap, and maximum jitter in sleep gap, as well as the number of quanta that need to be executed back-to-back within a wakeup cycle, which are used by QuarkOS to determine whether or not the task can be executed given current conditions.

Run-time system The QuarkOS run-time system is responsible for dynamically selecting how many quanta of a task to execute in a single wakeup interval, as well as the duration of sleep gap, while taking into consideration task requirements in terms of execution rate and jitter. The QuarkOS run-time builds on Dewdrop [4] to adapt to harvesting conditions, but addresses two limitations to enable it to scale-down to more constrained settings. First, QuarkOS takes the sleep-wakeup transition into account to minimize the overall overhead of transitions — this is not a problem for Dewdrop, which does not fragment tasks, but becomes an issue when each task is fragmented into several hundreds of quanta. Second, QuarkOS avoids ADC costs and only uses information from a threshold detector that detects when the voltage drops below a threshold that is considered high risk for outage. Instead of tracking operating voltage, it tracks the time after a threshold interrupt that it needs to wait before executing a quanta. Finally, the QuarkOS runtime also continually tracks gap sizes and jitter, and uses this to decide under what conditions to execute a task.

4 Preliminary results

We now present some initial results of our fragmentation-based network and image sensing stacks.

Bit-by-bit data transfer Figure 3 compares the bit-by-bit transfer capability of QuarkOS against a packet-based transfer scheme (4 byte packets) on the UMass Moo platform. We power the CRFID with an RFID reader and increase the transmit power levels. We constrain the device by replacing the default 10uF capacitor with a 0.1uF capacitor. Note that even though the buffer has less energy than required to transmit an entire packet, packet transfer can still work under sufficiently high harvesting conditions.

There are two interesting observations from the results. First, QuarkOS can operate at harvesting power levels that are a third of the minimum operating conditions for the packet-based transfer. In fact, QuarkOS's bit-by-bit transfer only fails when harvested energy is smaller than the system's lowest sleep power requirement. Second, it shows the adaptive capability of the QuarkOS controller, which increases the number of quanta per wakeup cycle as harvesting rates increase, thereby amortizing sleep overhead. Even under high harvesting conditions, the adaptive approach achieves close to the performance of the non-adaptive packet-based transfer mechanism.

Pixel-by-pixel image capture We implemented and evaluated the pixel-level image sensing using a low power camera (Stonyman [1]) and a CRFID platform (Moo) as the sensor controller. The whole image sensing platform is powered by a 3cm×3cm array of 9 solar cells under natural indoor light. We compare pixel-by-pixel capture in QuarkOS against a full-image capture scheme, where system starts image sensing when harvested voltage is higher than a predefined threshold (3.3V, the image sensor's operating voltage). Figure 2 shows a static 112×112 image captured by QuarkOS at a slow rate of about 1 image per minute. In contrast, the full-image capture scheme doesn't even get a chance to capture a single pixel because the system dies prior to this point.

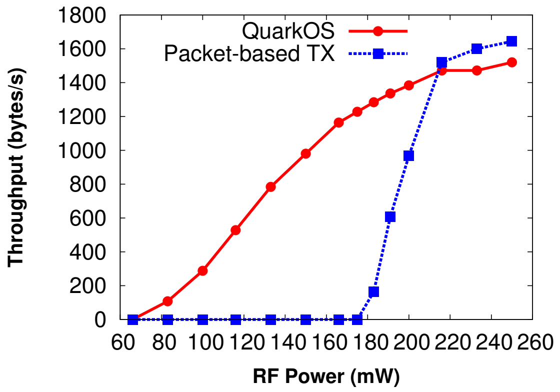

Figure 3: Throughput of bit-by-bit packet transfer.

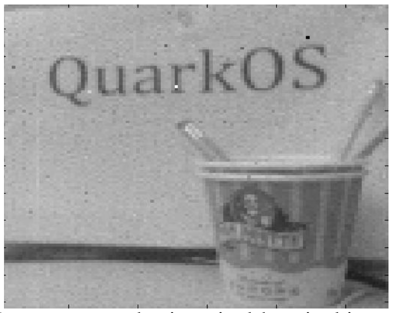

Figure 4: Image captured using pixel-by-pixel image capture.

5 Conclusion

In this paper, we present a simple but powerful abstraction, QuarkOS, that can enable systems to seamlessly scale down to miniature buffer sizes as well as ultra-low and dynamic harvesting conditions. Our approach is to deconstruct every task within the operating system and look for opportunities to insert sleep gaps to allow for the buffer to replenish before continuing execution of the task. Results show that our QuarkOS provides substantial benefits in pushing the limits of operation of micro-powered devices, and allow them to perform useful work under more extreme environments than previously thought possible. We believe that designing an operating system that works under such conditions can make it valuable to a wide range of emerging micro-powered embedded systems and applications.

References

[1] Stonyman Vision Chip Breakouts. `http://centeye.com/products/stonyman-vision-chip-breakout-board/`.

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

[3] S. Bandyopadhyay and A.P. Chandrakasan. Platform architecture for solar, thermal and vibration energy combining with mppt and single inductor. In *VLSI Circuits (VLSIC), 2011 Symposium on*, pages 238–239. IEEE, 2011.

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

[5] K.R. Carter, J.P. Rothstein, J.J. Watkins, T.P. Russell, T.J. McCarthy, L.J. Guo, M. Pasquali, B. Anthony, and D. Hardt. Roll-to-roll nanoimprint test bed. Center for Hierarchical Manufacturing, 2012.

[6] K. Dantu, B. Kate, J. Waterman, P. Bailis, and M. Welsh. Programming micro-aerial vehicle swarms with karma. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 121–134. ACM, 2011.

[7] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.

[8] A. Eswaran, A. Rowe, and R. Rajkumar. Nano-rk: an energy-aware resource-centric rtos for sensor networks. In *Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International*, pages 10–pp. IEEE, 2005.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ACM ASPLOS-IX*, 2000.

[10] Y. Lee, G. Kim, S. Bang, Y. Kim, I. Lee, P. Dutta, D. Sylvester, and D. Blaauw. A modular 1mm¡ sup¿ 3¡/sup¿ die-stacked sensing platform with optical communication and multi-modal energy harvesting. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 402–404. IEEE, 2012.

[11] P.D. Martin, Z.M. Charbiwala, and M.B. Srivastava. Doubledip: Leveraging thermoelectric harvesting for low power monitoring of sporadic water use. In *Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems*, SenSys '12, 2012.

[12] J.R. Polastre. *Design and implementation of wireless sensor networks for habitat monitoring*. PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California, 2003.

[13] Benjamin Ransford, Jacob Sorber, and Kevin Fu. Mementos: System support for long-running computation on RFID-scale devices. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '11, Newport Beach, CA, March 2011.

[14] A. Yakovlev, S. Kim, and A. Poon. Implantable biomedical devices: wireless powering and communication. *Communications Magazine, IEEE*, 50(4):152–159, 2012.

[15] D.J. Yeager, J. Holleman, R. Prasad, J.R. Smith, and B.P. Otis. Neuralwisp: A wirelessly powered neural interface with 1-m range. *Biomedical Circuits and Systems, IEEE Transactions on*, 3(6):379–387, 2009.

[16] 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, UMass Amherst Department of Computer Science, 2011.