Poster Abstract: Networked Embedded System Platforms in the Post-Mote Era

Pat Pannuto†, Michael P Andersen‡, Tom Bauer§, Bradford Campbell†, Amit Levy§, David Culler‡, Phillip Levis§, and Prabal Dutta†

†Computer Science & Engineering
University of Michigan
Ann Arbor, MI 48109
{ppannuto,bradjc,prabal}@umich.edu

‡Computer Science & Engineering
University of California, Berkeley
Berkeley, CA 94720
{m.andersen,culler}@berkeley.edu

§Computer Science & Engineering
Stanford University
Stanford, CA 94305
{tbauer01,levya,pal}@stanford.edu

Abstract

For the last fifteen years, research explored the hardware, software, sensing, communication abstractions, languages, and protocols that could make networks of small, embedded devices—motes—sample and report data for long periods of time unattended. Today, the application and technological landscapes have shifted, introducing new requirements and new capabilities. Hardware has evolved past 8 and 16 bit microcontrollers: there are now 32 bit processors with lower energy budgets and greater computing capability. New wireless link layers have emerged, creating protocols that support rapid and efficient setup and teardown but introduce novel limitations that systems must consider. The time has come to look beyond optimizing networks of motes. We look towards new technologies such as Bluetooth Low Energy, Cortex M processors, and capable energy harvesting, with new application spaces such as personal area networks, and new capabilities and requirements in security and privacy to inform contemporary hardware and software platforms. It is time for a new, open experimental platform in this post-mote era.

1 Introduction

Over a decade ago, a flurry of hardware platforms and supporting software empowered the research community to investigate and explore wireless sensor networks and their applications. Many projects today still use these "mote"-class devices, exploring problems in systems software, low-power networking, and application design. The underlying technology and its potential applications have progressed a great deal in the past decade. The Cortex M series of 32-bit processors finally have sleep currents competitive with MSP430 and AVR microcontrollers (MCUs).[1] 802.15.4 has grown far beyond the closed world of ZigBee, with new physical layers for new applications. The recent incorporation of Bluetooth Low Energy (BLE) into mobile phones allows ubiquitous sensing networks to directly interact with human-centric devices. Also driven by phones, sensors themselves are orders of magnitude more energy efficient and precise.

Simultaneously, applications have become much richer. Applications in early sensor network research focused on fixed rate, long-term sensing, guiding a research agenda of ultra-low power operation and robust multi-hop networking. Today, personal area networks (PANs), tether to phones and interact with proximity networks such as iBeacon; building-area networks share knowledge, like occupancy, among security, HVAC, and lighting control. In addition, the rise of "maker culture" and their platforms and communities (e.g. Arduino [1]) has led to a level of diversity and accessibility that early research platforms simply could not provide.

We have an explosion of new applications and developers, each with new and challenging requirements. We have reached a turning point in hardware, enabling a whole new class of device and operational models. It is time for a new OS and family of hardware platforms to explore and research embedded networked systems in the post-mote era.

2 Technology Today

Advancements in three major areas—networking, MCUs, and energy harvesting—open new opportunites for networked embedded systems and generate new implications on how an OS should manage activity, security, and privacy.

Integrating TinyOS and Bluetooth Low Energy. BLE provides highly efficient beaconing, bidirectional communication, fast device discovery, authentication, and optional piconetwork formation and management [3]. BLE explicitly incoporates ideas of periodic beacons and duty cycles, such that a device with a wakeup latency of a second can have an average current draw of < 40 µA.[2] Tight timing requirements mean that most chips have tightly integrated software stacks that present limited abstractions such as getting and setting attributes. Furthermore, these software stacks assume a threaded, sequential execution model that in our experience is hard to adapt efficiently to an event-driven paradigm. Finally, the current BLE model of embedded device coupled with a mobile app is built in the image of an application layer gateway, rather than an end-to-end addressable network elements, greatly limiting the application space.

[1] e.g. the NXP LPC11U6X family draws 1 µA in sleep with active RTC.

[2] e.g. Nordic Semiconductor NRF8001.

Submitted to: SenSys'14 November 3–6, 2014, Memphis, TN, USA

Cortex M0–M4. "Low-power processors" now includes highly performant cores such as the 1 μA–30 mA, 0.5–120 MHz Atmel SAM4S with a 32 bit Cortex M4, 2 MB flash, and 160 kB SRAM. Cm-scale chips now support primitives such as memory segmentation and dynamic frequency scaling. This presents an opportunity to enhance the robustness and capability of embedded OSes, if techniques such as memory protection and process isolation can be adapted to the embedded domain of tight hardware/software coupling.

New chips are evolving beyond the traditional definiton of MCU. More diverse and capable peripherals, such as encryption co-processors or even pseudo-TPMs (e.g. ARM TrustZone), are incorporated on-die as semi-integrated peripherals. While they often hang off a shared bus, these peripherals can be separately powered, programmed, and can even communicate with one another while the CPU core is asleep. Emerging platforms must provide expressive methods to leverage the growing set of capable heterogenous resources beyond the CPU, both on- and off-die.

Impact of Energy Harvesting. A decade of research in energy-harvesting transducers, efficient power electronics, and compact energy storage technologies, coupled with reductions in the idle and active power of radios, instant-on/instant-write phase-change memory, and the efficacy of modern MCUs make compact energy-harvesting sensors that can run perpetually, even in indoor settings, a reality [5]. However, writing applications for intermittently-powered, energy-harvesting systems is a challenge. It requires revisiting many assumptions about system startup, maintaining state across activation cycles, discovering and communicating with neighbors, predicting future energy availability, and scheduling operations under energy uncertainty [2]. Deriving clearer power state and availability semantics, and a means to express what state needs to be saved, present new embedded OS challenges.

3 Applications Today

A decade ago, stationary fixed rate (or occasionally event-driven) data reporting applications dominated sensor networks and their research. New applications that incorporate mobility, hybrid networking, and personal privacy are emerging. A new platform should embrace and enable these new opportunities.

Human-centric. The mobile device is a gateway to an embedded network. Instead of sending data to a fixed collection point, the network can send directly to a mobile device, on demand. This enables interaction with surrounding devices and infrastructure without needing to know URLs or logging into a cloud system. Proximity networks provide a degree of basic, physical security that enables opportunistic interactions.

Because these interactions are human-centric, they require low latency and are highly bursty – two performance properties unimportant in the dominant network model of the past decade. These low-latency interactions are more than event notifications or alarms. They can involve significant queries and exchanges of data. Furthermore, because they involve mobile devices, existing approaches of long-term link estimation are of limited use. These new application requirements require new communication abstractions, rethinking the trade-offs between latency, storage, energy, snooping, and how an OS will support them.

Perpetual Networks. Energy harvesting and low-power peripherals will finally enable a long-term goal of sensor networks: perpetual networks. Imagine iBeacons and smoke detectors that need no battery replacements. If the world will be filled with thousands of smart objects per person, energy must recede to be a non-issue for almost all of them.

We cannot predict the performance of a solar cell. This means an OS is stuck between two big unknowns: the future energy available as well as the potential energy needs from bursty, human-centric interactions. For an embedded device to be truly perpetual, there must be platform support to scale behavior and performance based on these two factors, optimizing needs, wants, and energy use.

Privacy and Proximity in Networks. Interactions between PANs (centered around a users mobile phone) and proximity applications such as iBeacons happen in public, with never-seen-before peers. This presents a dual security and privacy problem. On the one hand, connections between the PAN and proximity device must be confidential and authentic – e.g. payments. On the other hand, casual interactions with proximity devices must not enable ubiquitous tracking of users. Unfortunately, confidential and authentic communication and anonymity are difficult to achieve simultaneously. Operating systems can play a role in addressing these issues, for example by coordinating security features in the BLE stack with application specific knowledge.

4 A New Platform, Hardware and Software

Time and experience have identified fundamental concepts to re-think and re-build for the next generation embedded operating system [4]. Some key improvements we believe are necessary to focus on are: i) improving the developer experience, potentially incorporating non-traditional design tools to assist programmers, ii) adding advanced constructs such as C#'s await that enables sequential reasoning in an event-driven model, iii) developing better "operating system"-like behavior, better isolating application developers from resource contention and exposing interfaces that permit higher-level application logic to be written in friendly, easier languages (e.g. a syscall interface), iv) exploring new ways of programmatically conceptualizing time to enable network and system performance to better match programmer desire to system capability, v) expanding the operating system (or libraries) to provide more simple, less flexible, and more feature complete utilities like networking, without sacrificing the research community's capability to explore, and vi) deepening community and industry involvement, starting with building stronger ties and buy-in from people already invested in the embedded ecosystem and then looking towards enabling usability by true novices, much as Arduino and other upcoming platforms do today.

5 References

[1] Arduino. http://www.arduino.cc/. Accessed 28-July-2014.

[2] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: An energy-aware runtime for computational RFID. NSDI '11.

[3] C. Gomez et al. Overview and evaluation of Bluetooth Low Energy: An emerging low-power wireless technology. *Sensors*, 12(9), 2012.

[4] P. Levis. Experiences from a decade of TinyOS development. OSDI'12.

[5] P. Martin et al. DoubleDip: Leveraging thermoelectric harvesting for low power monitoring of sporadic water use. SenSys '12.

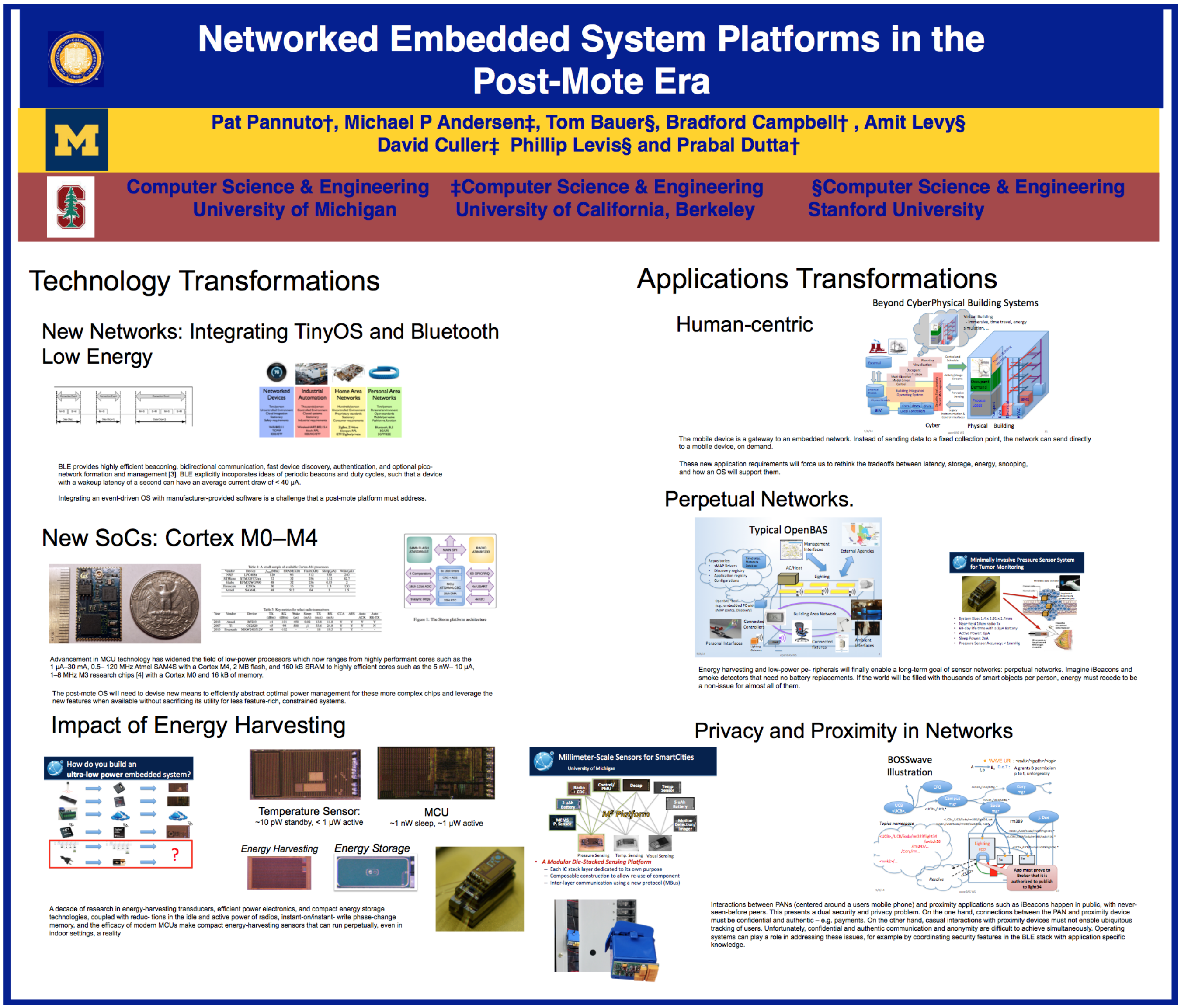

Figure 1: Poster Thumbnail