Mica: A Wireless Platform for Deeply Embedded Networks

Low-power integration of sensing, communication, and computation requires a new approach to wireless design. Flexible interfaces and primitive accelerators enable aggressive system-level optimizations.

Jason L. Hill
David E. Culler
University of California,
Berkeley

●●●●● Current wireless systems only scratch the surface of possibilities emerging from the integration of low-power communication, sensing, energy storage, and computation. Generally, when people consider wireless devices they think of such items as cell phones or personal digital assistants, items with high costs and energy requirements that target specific, highly standardized applications and rely on a powerful infrastructure. A new direction in wireless system design, however, is extending wireless connectivity to small, low-cost embedded devices for a wide range of applications.

Application possibilities for miniature wireless sensing devices include inventory asset tracking, roadside traffic pattern and open parking spot detection, individual plant monitoring for precision agriculture, habitat monitoring in nature preserves, and advanced building security and automation. The military could blanket fields with sensors to detect troop movement. Sensors might enable civil engineers to gauge the structural integrity of buildings and bridges after earthquakes or fires. Integrating hundreds of thousands of sensing and control points could provide new insights into the state of the world.

Exploiting local communication and application-specific protocols can drastically reduce size, cost, and power use in wireless devices. These devices won't need to communicate with the nearest high-power control tower, but only with their local peers. Peer-to-peer networking techniques provide a flexible mesh-like interconnect that shuttles data between thousands of tiny embedded devices. A handful of the devices might act as bridges between a local embedded communication mesh and a traditional data network. Figure 1 depicts a precision agriculture deployment—an active area of application research. Researchers are developing new algorithms for data aggregation, ad hoc routing, and distributed signal processing for low-power peer-to-peer wireless networks. As researchers envision smaller and lower-cost devices, the range of application scenarios grows dramatically.

The Mica wireless platform serves as a foundation for the emerging possibilities. Pictured in Figure 2, the Mica platform measures 1.25 × 2.25 inches, runs the TinyOS operating system (http://webs.cs.berkeley.edu/tos), and is suited for self-configuring multihop wireless networks. With sensing, communication, and I/O capabilities, Mica can simultaneously act

Figure 1. Ad hoc, wireless embedded network for precision agriculture. Sensors detect temperature, light levels, and soil moisture at hundreds of points across a field. The system communicates the data over a multihop network for analysis.

as a data router, sensor interface, and control point. Nearly a hundred research groups currently use Mica nodes to explore networking techniques, data analysis, distributed algorithms, networked services, programming, and novel applications. We created Mica with off-the-shelf hardware, but the architecture and its capabilities represent what could be implemented in just a few square millimeters of custom silicon. Mica's flexible design serves as a building block for creating efficient application specific protocols. Instead of defining narrow, standardized application interfaces, Mica provides a set of richly interconnected primitives (such as data serializers and timing extractors) to facilitate cross-layer optimizations. To explore novel systems approaches, researchers can develop customized protocols tailored to their application; Mica does not require use of predefined protocols.

Conventional wireless design comparison

An explanation of the current wireless architectures and their shortcomings for deeply embedded devices illustrates the advantages of the Mica network architecture. Deeply embedded wireless networks differ in several key respects from traditional wireless scenarios.

Figure 2. Mica node.

First, power consumption must be drastically reduced. A deeply embedded, battery operated device might need to operate for years on a pair of AA batteries or a lithium coin cell. Power consumption must average in the microamps range, requiring powering down most of the device much of the time—a cell phone would do well to last a couple of weeks
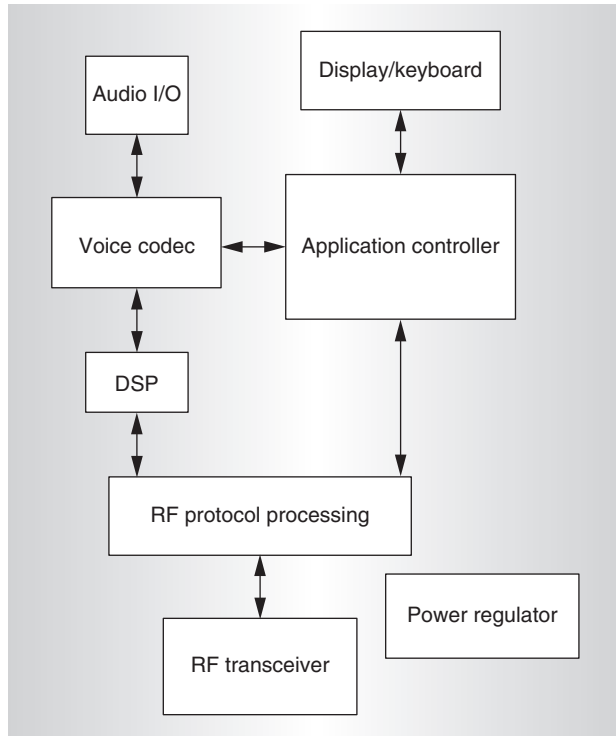
Figure 3. Typical cell phone architecture. The design is partitioned into highly specialized pieces, limiting new application scenarios.

on two AA batteries. This hundredfold power consumption reduction mandates drastic changes to system architecture. Second, many deeply embedded systems cannot rely on a predeployed, powerful infrastructure for support. Cell phone protocols, for example, exploit high-powered base stations to reduce power consumption in mobile nodes. In a peer-to-peer multihop network, typically, only the last hop will communicate with a base station—most of the nodes would be located outside of the stations direct communication range. An advantage of a peer-to-peer architecture is that it only requires small communication distances for each transmission. Finally, whereas most wireless devices carry out a single, highly standardized function, deeply embedded networks must be suited to a wide variety of applications. This requires a more general-purpose system design, but also allows employing radical, highly efficient algorithms.

As wireless transmission power decreases, the significance of protocol processing increases. An average cell phone uses 750 to 1000 mW to reach a distant cell tower. In such

devices, protocol processing can consume hundreds of milliwatts before making significant battery life impact. In a deeply embedded wireless network, however, devices typically require less than a milliwatt to communicate with their neighbor nodes. This drastic reduction in transmission power must be complemented by a highly efficient protocol processing mechanism. In wireless Ethernet (IEEE 802.11b) cards, radio-frequency (RF) transmission power ranges from just 25 to 100 mW, yet device power consumption can exceed 2,000 mW when active due to protocols that are optimized for high data rates. A pair of AA batteries would last less than eight hours with a standard 802.11 card. Thus, in deeply embedded wireless networks, protocol processing must be optimized for ultralow-power operation, not for high data rates.

Traditional and deeply embedded wireless devices also differ in their required level of flexibility. Peer-to-peer protocols and in-network processing implementations must adapt to meet application needs in sensor networks. Currently, cell phones use highly partitioned designs; a generic microcontroller provides the user interface and device configuration, dedicated digital signal processors process audio with a single and highly specific algorithm, and complex radio controllers provide the low-level channel processing. Figure 3 depicts a generalized cell phone architecture. A prolonged standardization process allowed for high optimization for specific functions in each system component, with narrow interfaces between components. New services are typically overlaid on the established structure. For example, on second-generation cell phone systems, a point-to-point layer over what is essentially an audio stream provides an IP data communication stack. The 802.11 designs are also highly partitioned, each of the components service a specific, highly standardized aspect:

- A base-band controller implements the radio channel interface (including spectrum sequencing, framing, and coding);
- A bus controller services the PCMCIA interface; and
- A microcontroller shuttles data units between these controllers according to a specific media access control (MAC) protocol.

This strict partitioning makes it difficult for devices to adapt their behavior to meet application-specific requirements. Additionally, the complex controllers surrounding the radio consume significant power bringing the radio online making low-power operation difficult.[1]

The Bluetooth wireless specification targets low-power personal-area wireless communication. Originally envisioned to replace the wires of modern PCs—a wireless USB—the Bluetooth design is based on a master-slave model in which a single master device communicates with a few peripherals (less than eight). While the specification lets devices participate in logical communication groups distinct from the primary master-slave relationship, it has a powerful master in direct control of a small collection of physically close low-power devices. In the deeply embedded networks we envision—a mesh of thousands of interconnected embedded nodes—the key to power, cost, and size reduction is neighbors forwarding and routing data. Ad hoc, multi-hop interconnections could be simulated as an overlay on the Bluetooth master-slave physical layer, but are likely to be inefficient. Additionally, the Bluetooth device power consumption is still an order of magnitude above our intended target—Bluetooth chipsets consume 115 mW to communicate to a master node.[2] Similar to IEEE 802.11 devices, current Bluetooth designs are partitioned to implement a single rigid protocol in conjunction with a well-powered host device, such as a laptop or, eventually, a personal digital assistant.

Exploiting wireless design

For the deeply embedded approach, it's essential that researchers explore radical directions in system-level optimization. Cell phones, wireless local area networks, and Bluetooth protocols must meet strict bandwidth and latency requirements. Cell phone voice traffic transmissions cannot suffer signal delays that cause noticeable audio variations. Additionally, a small transmission round-trip time between users is required to keep an interactive conversation flowing smoothly. These demands require that these devices meet strict protocol design specifications. In deeply embedded wireless networks, however, we can exploit trade-offs between bandwidth, latency, and in-network processing to reduce power consumption by orders of magnitude. For example, if sensor data is sampled only once per minute, it might be acceptable to delay transferring the data to the consumer for several seconds, allowing the network to coordinate many such flows efficiently while operating at a low duty cycle. Such optimizations can extend battery life from weeks to years.

Our deeply embedded system approach emphasizes flexibility in ultralow power operation and the opportunity to produce system-level optimization. Rather than a narrow, standardized interface to a complex radio controller, our deeply embedded designs use simple radios with much of the RF channel control exposed to software through a rich interface. Rather than dedicated protocol controllers, we provide simple accelerators for protocol primitives that are composed in software. Rather than partitioning the design into dedicated subsystems, this new approach pools processing resources and uses fine-grain multithreading to dynamically allocate sub-task processing. While each of these differences affects performance on simple tasks, their real impact is made by enabling high-level software optimizations.

Mica design

We designed the Mica platform to aid system level exploration, providing a rich interconnection of protocol primitives that can designers can flexibly assemble into application-specific protocols. Our design's power and time efficient primitives significantly improve key system capabilities including low-power device start up, time synchronization, power-aware routing, and localization.

Mica combines communication, computation, power management, and sensing into a small experimental platform. The current form factor (1.25 × 2.25 inch) is a similar size as a pair of AA batteries, although, we have compressed a variant of the design to about the size of a 2.5-centimeter coin (0.5 cm thick). The standby current for Mica's components is a few microamps, enabling applications that last for several years on a single set of batteries. Mica improves experimental flexibility by including an expansion bus that connects to a wide array of sensor boards. Our current batch of sensor boards includes support for monitoring ther-
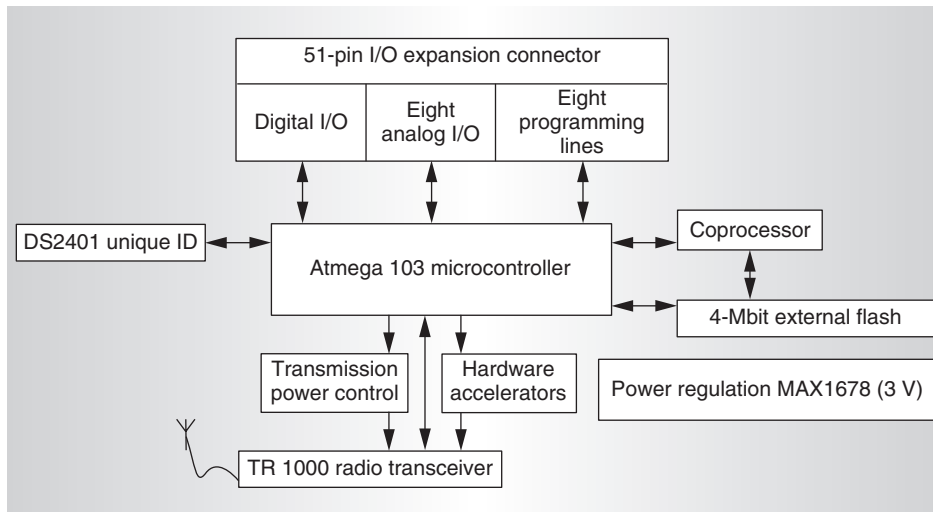
Figure 4. Mica architecture block diagram. The direct connection between application controller and transceiver enables flexibility in meeting application demands. Hardware accelerations optionally assist in communication protocols.

mal temperature, barometric pressure, magnetic fields, light, passive infrared, acceleration, vibration, and acoustics.

Block diagram overview

Figure 4 shows the Mica architecture, consisting of five major modules: processing, RF communication, power management, I/O expansion, and secondary storage. A quick survey of the major modules provides a general overview for the system as a whole, hence, a detailed bill of materials, device schematic and datasheet for all components mentioned can be found at http://www.tinyos.net.

The main microcontroller is an Atmel ATMEGA103L or ATMEGA128 running at 4 MHz and delivering about four million instructions per second (MIPS). This 8-bit microcontroller has

- a 128-Kbyte flash program memory,
- a 4-Kbyte static RAM,
- an internal 8-channel 10-bit analog-to-digital converter,
- three hardware timers,
- 48 general-purpose I/O lines,
- one external universal asynchronous receiver transmitter (UART), and
- one serial peripheral interface (SPI) port.

Normally, programming of these embedded microcontrollers occurs during manufacture with a firmware upload or during field maintenance. In our design, however, the embedded network can be dynamically reprogrammed during routine use. The coprocessor handling the wireless reprogramming is an Atmel AT90LS2343, 8-pin, flash-based microcontroller with an internal system clock and five general-purpose I/O pins. Additionally, to provide each node with a unique identification, we include a Maxim DS2401 silicon serial number—a low-cost ROM device with a minimal electronic interface and no power requirements.

The Mica radio module consists of an RF Monolithics TR1000 transceiver and a set of discrete components to operate the radio. Software can externally set the transceiver's transmission radius to range from inches to hundreds of feet, and it operates at communication rates up to 115 kilobits per second. This amplitude-shift-keying based radio has a fixed transmission frequency of 916.5 MHz. At maximum transmission power, it outputs approximately 0.75 mW—roughly 1/1000 the power of a cell phone—and consumes 21 mW. With a maximum receive sensitivity of less than 95 dBm, or decibels relevant to one milliwatt, it provides an unobstructed communication range of approximately 200 feet. In receive mode, the radio consumes 15 mW regardless of whether actual communication is occurring. The radio interface gives direct control over the transmitted signal, allowing the modulation scheme, coding, framing, and MAC protocol to be determined in software. In addition, operating system software controls radio transmission strength and can sense the strength of the receive signal.

A 4-Mbit Atmel AT45DB041B serial flash chip provides persistent data storage. We chose this chip because of its interface and small footprint, 8-pin small-outline integrated circuit. It stores sensor data logs and temporarily holds program images received over the network interface. To hold a complete

program the flash must be larger than the 128-Kbyte program memory. This prevented the project from considering use of the lower power, electronically-erasable-programmable-ROM-based solutions because they are generally smaller than 32 Kbytes.

We designed the power subsystem to regulate the system's supply voltage; a Maxim1678 DC-DC converter provides a constant 3.3-V supply. Mica operates with inexpensive alkaline batteries that produce between 3.2 and 2 V (for example, a pair of AA batteries). We chose the Maxim chip because of its small form factor and high efficiency. The converter takes input voltage as low as 1.1 V and boosts it to 3 V. This supplies a clean, stable voltage source for the rest of the system. Input voltage significantly affects the TR1000 transmission strength and its receive sensitivity. The converter chip increases the system's available power because more than 50 percent of the energy in an alkaline cell lies below 1.2 V, which is unusable without a boost converter. For ultralow-power sleep mode, disabling the power system lets the system run directly off the unregulated input voltage, reducing power consumption by the boost converter and the microcontroller. The radio will not operate, however, without the boost converter enabled. Table 1 summarizes Mica's node component power consumption.

The I/O subsystem interface consists of a 51-pin expansion connector that we designed to interface with a variety of sensing and programming boards. We divided the connector into the following sections:

- eight analog lines,
- eight power control lines,
- three pulse width modulated lines,
- two analog compare lines,
- four external interrupt lines,
- an I2C-bus from Philips Semiconductor,[3]
- an SPI bus,
- a serial port, and
- a collection of lines dedicated to programming the microcontrollers.

The expansion connector can also program the device and to communicate with other devices, such as a PC serving as a gateway nodes. Additionally, it contains a standard UART interface to control or provide data to

Table 1. Breakdown of active and idle power consumption for Mica hardware.

Component	Active (mW)	Idle (mW)
CPU	16.5	30
Radio	21 (transmit mode)	0
	15 (receive mode)	
Silicon ID	0.015	0
External flash	45	30
LEDs	10	0

any RS-232-protocol-based device. Dozens of sensor boards with a variety of sensors have been developed that use this expansion connector. It has even been used to let the Mica node control a handful of inch-sized microrobotic platforms.

Operating system

The Mica hardware platform uses the TinyOS multithreading execution model developed at the University of California, Berkeley.[4] TinyOS is an event-based operating system in which individual components act together to form complete application and implement all system functions. This component-based structure lets an application designer select from a catalog of system components to meet application specific goals.

The TinyOS execution model provides fine-grained allocation of processing resources across multiple components. This lets high-level application code and low-level protocol code coexist on a single CPU. Each component acts like a finite state machine that uses commands and events to transition from one state to the next. There is no blocking or waiting in TinyOS. This forces components, after finishing a calculation, to release the CPU for use by other components.

TinyOS performs high-level, long-running application processing in special execution contexts, called *tasks*. When executed, a task runs to completion and other tasks cannot preempt it. However, low-level system events can preempt tasks, allowing TinyOS to temporarily reallocate the CPU to low-level system processing. TinyOS shields the application-level processing from the underlying concurrent scheduling, yet exposes low-level system components to meet their real-time requirements.

In contrast, most wireless devices use dedicated protocol processors to handle low-level processing. This partitioned approach requires enough power in the application and system-level processors to handle the peak demand; though system demand for these two processors might never occur simultaneously. The pooled processing approach used in TinyOS allows dynamically allocating a single CPU to a given task, so peak demand is close to average processing. For example, the radio packet start symbol detection requires a peak of 3 MIPS, but only at the instant a packet is received (less than one percent of the time). Suppose the signal processing algorithms to analyze sensor readings requires 2 MIPS each time a sample arrives. A partitioned system would require a total of 5 MIPS, however, using a shared pool approach, slightly more than 3 MIPS could handle this application. Additionally, in a real application with a mix of operations of varying cost, efficiency decreases further because of over provisioning of the partitioned system to the worst case. Furthermore, if a more efficient mechanism for performing start symbol detection was selected for a particular application scenario, TinyOS could reallocate the freed resource.

The biggest advance gained from exploiting a shared pool of computation is that the interfaces between system components are not constrained by narrow chip-to-chip communication mechanisms defined by physical hardware. System components can provide rich interfaces to other components to enable cross-layer optimization. Additionally, software-based interfaces facilitate interface evolution as new features are added to system components.

Raw radio interface

The radio subsystem is a prime example of the rich simple interface approach. From this subsystem's low-level interface, designers can build customized arbitrary signaling protocols to meet application requirements. Only the radio dictates certain parameters, that is, signaling should occur over a half-duplex bit-serial link using amplitude shift keying with a minimum physical bit time of 10 microseconds (μs), and the need for a rough direct current (DC) balance (for example, there should not be more than four consecutive high bits or four low bits). Beyond that, system software

components determine all aspects of the communication protocol. In contrast, Bluetooth radio chipsets dictate bit rates, signaling schemes, data-encoding methods, framing, MAC protocols, and the routing scheme to the host processor. With Bluetooth, the operating system can only spool packets across the interface; delays through the interface have a large, unpredictable variance, and the system or application components cannot observe the activity or physical characteristics of the radio channel. Researchers have observed similar shortcomings with IEEE 802.11 designs.[5] Figure 5 depicts a standard packet-based protocol implemented on the Mica node.

The TR1000 radio used on the Mica gives the controller direct access to the signal strength of the incoming RF transmission, and sampling the level of background noise during periods when there is no active transmission, and adjustment of transmission strength. Using this information in multihop networking applications can dramatically improve efficiency. Interactions between transmitter and receiver are very predictable, as is the delay through the radio interface. Because software can quickly and predictably turn radio power on or off, low duty cycle operation can occur without global coordination or complex time slotting. This direct, low-level interface to the radio provides flexibility for application developers. Researchers from the University of California, Los Angeles have exploited this flexibility to create energy aware MAC protocols.[6]

Communication accelerators

The drawback of a low-level interface to the radio is that it places a significant overhead on the main controller, because of the frequency of programmed I/O operations and the inefficiency of conventional instruction sets for expressing these operations. It's inefficient to handle the bit-serial sliding window correlation operation required for start symbol detection on a general-purpose data path, although this takes just a few gates to implement in hardware. The design of dedicated protocol processors typically contains support for such operations. To compensate for this, the Mica architecture includes hardware accelerators for the most demanding primitives used in protocol construction and allows these primitives
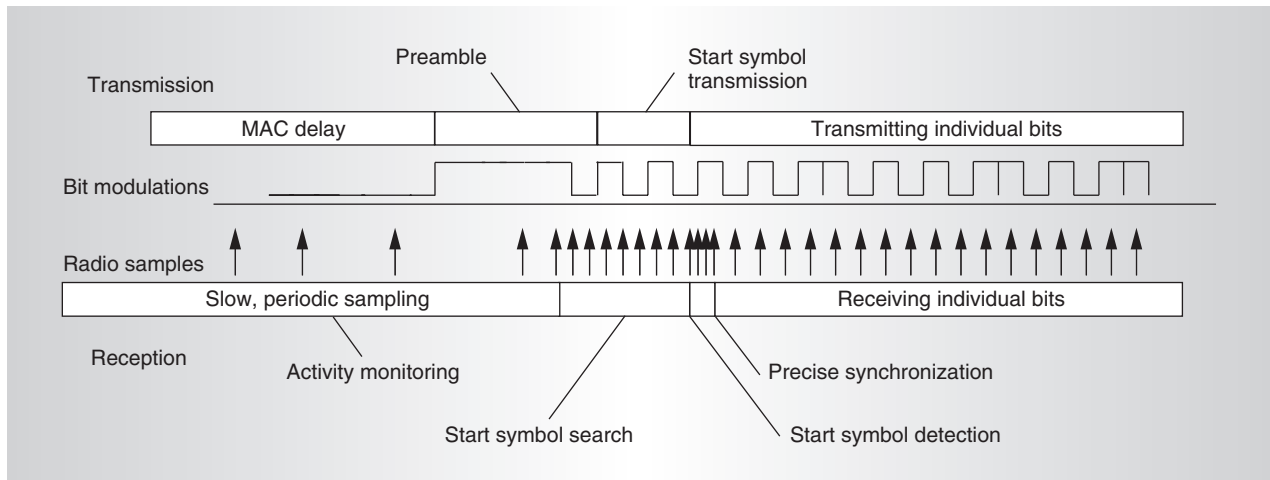
Figure 5. Anatomy of packet reception and transmission. For reception, slow sampling detects arrival of a 10-kilobits per second start signal. The packet's precise timing helps synchronize for the 50 kbps data payload. Data sample timing for the entire packet is based on the initial synchronization. For transmission, after a short random delay, the preamble, then a start symbol, and then the data are sent.

to be composed in software, making rich interfaces available. In general, these primitives could be implemented in board-level logic, field-programmable gate arrays, application-specific integrated circuits, or integrated logic, depending on the implementation technology.[7] For the Mica platform, we built the critical hardware accelerators using conventional serializers in unconventional ways. While they provide a significant efficiency boost, they represent just the beginning of the possibilities we can include as we progress toward custom devices.

The first accelerator provides a simple shared memory buffer between the bit-parallel data path and the bit-serial radio channel. This lets the processor deal with communication data in efficient chunks and overlap its data manipulation with the low-level spooling of bits to or from the radio, at defined intervals. The mechanism doesn't dictate channel coding or signaling and can be bypassed to let the microcontroller direct interaction with the radio by using programmed I/O. This option meets real-time requirements when necessary, for example, during start symbol detection, but doesn't force real-time processing of all transfer options. Software can deal with data in 8-bit chunks when optimal, but can also be directly accessed bit-by-bit as it arrives. This accelerator lets the Atmel controller drive the radio at 50 kbps using a small fraction of the

processor, whereas programmed I/O peaks at 20 kbps using the entire processor. Other microcontrollers with a double-buffered SPI transmit port can achieve the full 115 kbps of the radio. The byte streaming performance of more sophisticated versions of this accelerator would approach conventional direct memory access channel performance , yet with precise control over timing.

The Mica node also incorporates a synchronization accelerator that captures the exact timing of an incoming packet to within one clock cycle (250 nanoseconds) at the start of packet reception. Shared memory stores this packet time stamp and the data path can read it. This hardware accelerator is critical to our design because at high bit rates, determining transmission timing is extremely difficult. The synchronization accelerator forwards this time stamp to the buffering accelerator so that it can perform automatic channel sampling at the center of each bit transmission. The timing information is not buried in the radio interface; it can also be delivered to application software for use in higher-level synchronization operations.

We built both of these hardware accelerators out of standard microcontroller functional units. SPI is a synchronous chip-to-chip interconnect with a serial data line and a separate clock signal that an external SPI master will provide. We drive the asynchronous radio

by combining the functionality of input capture registers, timer controlled output pins, and the SPI communication hardware. The input capture register automatically captures a timing pulse contained in the packet. This value is used to configure the timing register that controls an output pin. One control option lets hardware automatically toggle the output pin each time the counter expires; this output pin becomes a clocking signal that times each arriving bit. Finally, SPI hardware captures the value of the radio signal each time the counter triggers the clock line. We accomplish this by connecting the counter controlled output pin to the synchronous clock line of the SPI port. On the Mica node, we combine the radio's incoming receive data with an artificially generated clock signal to create an SPI master. The clock signal is the output of the timing register fed back into the controller's SPI port. This results in the SPI port automatically latching and buffering the incoming transmission with precise timing—as long as the internal timing register is configured correctly at the start of reception.

Cross-layer optimizations

Many deeply embedded applications spend the vast majority of the time in a very low-power state, slowly draining their available energy supply by processing sensor readings to confirm that no particular action need be taken. Infrequently, these applications communicate monitoring results and, rarely, portions of the network become extremely active upon detection of an important event. Performance improvements in an application's active work, such as processing and messaging, enhance the application's overall effectiveness and let it return to a low-power state more quickly. However, the larger impact of the richly integrated approach to wireless design is that enables optimization of the rest of the application. Often these optimizations reach across traditional layers of abstraction, using low-level information to achieve a high-level goal.

RF wake up

Upon detection of an important event, a node will typically awaken, or start up, a substantial portion of the network. The network will collect, transport, and process data from several sources and communicate results rapidly to engage some response. This wake up would normally be realized above the messaging layer, passing packets to propagate the event notification to initiate aggregate processing. However, we can exploit the flexible interface to the radio to implement an ultralow-power network wake-up signal. More specifically, this flexibility lowers system cost for wake-up checking.

For any RF based wake-up protocol, each node must periodically turn on its radio and check for a wake-up signal. Consider a protocol in which a node checks for a signal every 4 seconds. Since the protocol assumes that the wake-up alarm is rare, the source can transmit a signal continuously for a period of time after the event. A packet interface uses the detection of a start symbol pattern to differentiate data from noise on the channel. Thus, to implement the wake-up check over a packet-based radio interface, a node must turn on its radio for at least the duration of two packets. We typically use 30-byte packets with a DC-balanced, single-error correction, dual-error detection encoding that results in 540 bits per packet. Thus, at 10 kbps, the check time would be 108 milliseconds (ms), or 21 ms at 50 kbps; just checking for wake up yields a 2.7 percent active duty cycle of the device, or 0.5 percent at the faster rate. Although, reducing the responsiveness to the alarm or modifying the packet interface for an early rejection on the check could reduce this cost, we perform a more radical cross-layer optimization.

Instead of interacting with the radio over a high-level packet interface, our low-power sleep mode interacts directly with the analog base band. The wake-up signal is nothing more than a long RF pulse. A node checks for the wake-up signal by sampling the energy on the channel and can determine whether the wake-up signal is present in less than 50 μs. With a four second check interval, this 2,000-fold speedup in the check time results in a 0.00125 percent radio duty cycle. To put this in perspective, the same energy consumed by packet-based checks in a week could last 38 years using low-level wake up. We have used this ultralow duty cycle implementation to consistently wake up multihop sensor networks of more than 800 nodes.

Time synchronization

Many sensor applications require time-correlated sensor readings or multimode coordination, and therefore need an underlying time-synchronization mechanism. For example, if a sensor network were used to capture the propagation of earthquake vibrations through a building, sensor readings across the network would need to be synchronized to within hundreds of microseconds. The accuracy of distributed synchronization protocols is bounded by the unpredictable jitter on communication times. Aside from variations in the actual network latency, the variation in delay going through sophisticated protocol processors can be hundreds of milliseconds dues to buffering, MAC protocols, and backoff. Unlike wide-area time synchronization protocols—such as network time protocol[8]—we can determine all sources of communication delay. Our use of rich interfaces lets us expose the sources of delay to the application, reducing the unknown jitter. Additionally, by exploiting shared system timers, we can assign precise time stamps to incoming and outgoing packets beneath the sources of variation.

We designed the Mica platform to use an internal 16-bit high-frequency counter to act as the lower 16 bits of a 32-bit continually running system clock. This highly accurate system clock is directly linked to the synchronization accelerator used to capture the exact timing of an incoming packet. To synchronize a pair of nodes, a packet can be time-stamped with a sender's clock value as it is transmitted and after all MAC delays have occurred (prior to transmission, a node might have to wait for the radio channel to be clear). The receiver's synchronization accelerator can then tag the packet with the receiver's clock value. The application can use the difference between the two time stamps to determine how to adjust its system clock.

Our implementation synchronizes a pair of nodes to within 2 μs. We can directly attribute the skew of +/- 2 μs to several sources of jitter. When sending, there is a jitter of +/- 1μs in the transmission propagation due to the internal circuit dynamics of the radio. Hardware then captures the arriving pulse with an accuracy of +/- .25 μs. Finally, to synchronize the clock based on the captured value introduces an additional +/- .625 μs of jitter. This imple-

mentation is only possible because of the shared access to a high-accuracy system timer between the bottom of the network stack and the top of the application. Partitioned wireless devices—such as Bluetooth chipsets—hide the exact timing information from applications. (Generally the receive packet path has less jitter than the transmit path, so an alternative approach uses the broadcast nature of the radio channel to synchronize multiple receivers, even if they are poorly synchronized with the transmitter.[9]) Mica's time synchronization primitives provide highly accurate pairwise synchronization between any two nodes. This primitive can aid construction of network-wide time synchronization to within tens of microseconds.[9]

Determining device location

Many applications need to associate location information with the data obtained from sensor nodes. Using the radio as both a sensor and a high-fidelity time synchronization mechanism can facilitate localization. RF signal strength falls off with distance, so sampling the strength of the base-band signal, combined with information on the transmit power level, provides an indication of distance. The distance estimates can aid an elaborate form of triangulation to determine device position.[5,10] Variations in strength readings due to multipath effects, interference, and transmitter/receiver tuning make this approach more effective for determining proximity than absolute position.

Alternatively, obtaining accurate time stamps can help determine the propagation delay of acoustic pulses as a basis for localization.[11] Some sensor boards contain an electronic sounder and a microphone. These boards transmit a radio packet in unison with an acoustic pulse. The RF packet causes the receiver to start a counter to measure the delay until the acoustic pulse arrives, providing calibration parameters to map measured delay to distance. This mapping compensates for delays in the acoustic transmission/receive path. As with the radio, a rich interface reduces the sources of unknown jitter.

Application evaluation

The ultimate goal of the Mica node is enabling novel applications that shape the role

Table 2. Node energy costs in a persistent data monitoring application that reports every 5 minutes and has a maximum of five children. The table shows results for basic and optimized implementations.

Implementation	Base	Optimized	Improvement factor
Reports/day	228	228	228
Samples/day	21,600	21,600	21,600
RF wakeup duty cycle (%)	2.82	0.02	—
Data communication (mJ/day)	2,262	452	5
Children's transmission waiting time (mJ/day)	669	33	20
Alarm check (mJ/day)	54,237	25	2,160
Sensing (mJ/day)	17	17	1
Total active (mJ/day)	57,187	528	108
Sleep (mJ/day)	5,038	5,182	1
Total (mJ/day)	62,225	5,711	11

of deeply embedded networks. Not only do the application requirements differ drastically from those of traditional wireless scenarios, they differ dramatically from one another. In our exploration of these new scenarios, two distinct classes of applications have emerged. The first is categorized by a low duty-cycle, low data-rate, long latency, static topology and a long expected lifetime. For these networks, lifetime is the main evaluation criterion. A second class of applications is that of highly dynamic sense-and-control networks with higher data rates, latency limits, and highly mobile nodes. Instead of passively monitoring a relatively static environment, these networks attempt to control the environment in real time. This class places strict latency and throughput requirements on the network, which act as the main measure of system performance.

A common low duty cycle scenario for sensor networks is the continual monitoring of an environmental space for low-frequency environmental changes combined with detection of important events. These scenarios include heating, ventilation, and air conditioning monitoring of large buildings or climactic monitoring of outdoor habitats. The Mica optimization opportunities come together to determine the application's capability and lifetime at a given energy budget. Consider a scenario where the network forms itself into a tree and each node samples environmental data every 4 seconds, aggregates a statistical summary over 5 minutes worth of readings, and transmits the summary to its parent node. Each parent combines the read-ings from its children and sends a single summary to its parent. In the event of a drastic environmental change, any node can send an emergency notification to its parent at any time. The parent checks for emergency messages every 4 seconds. Table 2 summarizes the theoretical impact of the MICA optimizations for a base case without the hardware accelerators and a case with the accelerators and the cross-layer optimizations that they enable. MICA's support of application specific protocols results in a 10 × increase in overall application performance.

The energy cost of data transmission assumes that the worst-case node is responsible for five children. The node must receive a summary from each of the children, combine them, and transmit a new summary to its parent. The processing time is small compared with the time needed to transmit or receive a message. The hardware accelerators reduce— proportionally to the increase in bandwidth—the energy consumption of this portion of the application.

Since communication is infrequent and the radio consumes energy whenever it's powered on—whether it's receiving data or not—nodes leave their radios powered off whenever possible. In a rendezvous protocol, nodes agree to communicate at a future time and turn off their radios until that time. In these protocols, synchronization precision translates into lower power consumption. If a scheduled application-level rendezvous is only accurate to approximately 100 ms, then the receiver must wake up 100 ms early to ensure that it's not late; this could result in the receiver waiting

200 ms before the transmission actually begins. For the short transmissions common in sensor networks, this results in a four to five times increase in cost. The use of bit-level time synchronization reduces the energy consumed waiting for the rendezvous by a factor of 20.

The relatively high frequency of the wake-up check makes it the single largest consumer of energy in the base design. Exploiting the low power wake-up functionality reduces this cost by a factor of 2,000. Taken together, the three optimizations (RF wake up, time synchronization, and high-speed communication) reduce energy consumption of the active part of the application by two orders of magnitude. Table 2 shows the impact of each individual optimization as well as the overall impact on system lifetime for a theoretical deployment. In the base case, the power drain in sleep state was a reasonable 10 percent of the overall budget, but it dominates the optimized application. When looking at the application as a whole, battery life is improved by a factor of 10, resulting in an expected lifetime of 10 years for a pair of AA batteries (that is, 9 years for the base and 1 year for the optimized implementation). Fortunately, further integration of the design will further reduce the standby power drain.

The application must control the duty cycle and rendezvous schedule of the protocol, because these elements vary greatly depending on the application scenario. For example, temperature sensing might require 1-minute intervals between reports with only a byte of data collected per node, while network monitoring of plant photosynthesis could need more complex readings, but only once per hour.

The exploration of deeply embedded sensor networking is pushing wireless technologies in new directions. Many research groups in the US that are exploring these directions use the Mica node as a foundation for their research. A Single-chip version of the architecture is in development today. It will generalize Mica's architecture, provide additional accelerators, and dramatically reduce the standby power consumption. **MICRO**

Acknowledgments

References

1. M. Stemm et al., "Reducing Power Consumption of Network Interfaces in Hand-Held Devices," IEICE Trans. Comm., vol. E80-B, no.8, 1997, p. 1125-31.

2. TC2000 Single Chip Bluetooth Solution, Zeevo, Santa Clara, Calif., 2001; http://www.zeevo.com/pdf_files/tc2k_public.pdf.

3. I²C-Bus Specification v2.1, Philips Semiconductor, Eindhoven, Netherlands, 2000; http://www.semiconductors.philips.com/acrobat/various/I2C_BUS_SPECIFICATION_3.pdf.

4. J. Hill et al,, "System Architecture Directions for Networked Sensors," Proc. 9th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), ACM Press, New York, 2000,

5. P. Bahl and V. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," Proc. Ann. Joint Conf. IEEE Computer and Comm. Soc. (IEEE Infocom), vol. 2, 2000, IEEE Press, Piscataway, N.J., pp. 775-784.

6. W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," Proc. Ann. Joint Conf. IEEE Computer and Comm. Soc. (IEEE Infocom), IEEE, Piscataway, N.J., 2002.

7. J.L. Da Silva, Jr. et al., Design Methodology for Pico Radio Networks, Berkeley Wireless Research Center, Berkeley, Calif., 2001.

8. D.L. Mills, , "Internet Time Synchronization: The Network Time Protocol," IEEE Trans. Comm., vol. COM-39, no. 10, 1991, pp. 1482-1493.

9. J. Elson and D. Estrin, "Time Synchronization for Wireless Sensor Networks," Proc. IPDPS Workshop Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, Springer-Verlag, Germany, 2001.

10. J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," Computer, vol. 34, no. 8, 2001, pp. 57-66.

11. N.B. Priyantha, , A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support

System," Proc. 6th Ann. Int'l Conf. Mobile Computing and Networking (MobiCom), ACM, New York, 2000, pp. 32-43.

Jason L. Hill is currently working towards his PhD at UC Berkeley. His research interests include developing hardware and software to enable the proliferation of wireless sensor networks and developing millimeter-scale wireless sensor nodes by incorporating communication, computation, and storage onto a single CMOS chip. Hill has a BS and an MS in electrical engineering and computer science from the UC Berkeley.

David E. Culler is a professor of computer science at UC Berkeley and founding director of Intel Research, Berkeley. His research interests include vast networks of small, embedded wireless devices; parallel computer architecture; parallel programming languages; high-performance communication; TinyOS; networks of workstations (NOW); Internet services; active messages; split-C, the threaded abstract machine; and data flow systems. Culler has a BA from UC Berkeley, and an MS and PhD from the Massachusetts Institute of Technology. He was awarded the NSF Presidential Young Investigator in 1990 and the Presidential Faculty Fellowship in 1992, and holds three patents.

Direct questions and comments about the article to Jason Hill, 467 Soda Hall Berkeley, Calif. 94720-1776; jhill@cs.berkeley.edu.

For further information on this or any other computing topic, visit our Digital Library at http://computer.org/publications/dlib.