

CS 598IG
 Adv. Topics in Dist. Sys.
 Spring 2006

Indranil Gupta
 Lecture 4
 January 26, 2006

Hype, But do we need this new technology?

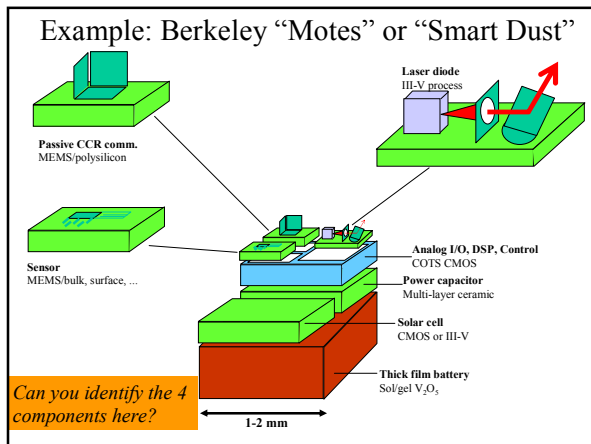
- Coal mines have always had CO/CO2 sensors
- Industry has used sensors for a long time
- Today...
 - Excessive Information
 - Environmentalists collecting data on an island
 - Army needs to know about enemy troop deployments
 - Humans in society face information overload
 - Sensor Networking technology can help filter and process this information (And then perhaps respond automatically?)

Growth of a technology requires

- I. Hardware
- II. Operating Systems and Protocols
- III. Killer applications
 - Military and Civilian

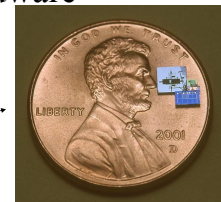
Sensor Nodes

- Motivating factors for emergence: applications, Moore's Law, wireless comm., MEMS (micro electro mechanical sensors)
- Canonical *Sensor Node* contains
 1. Sensor(s) to convert a different energy form to an electrical impulse e.g., to measure temperature
 2. Microprocessor
 3. Communications link e.g., wireless
 4. Power source e.g., battery

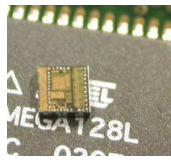


Example Hardware

- Size
 - Golem Dust: 11.7 cu. mm
 - MICA motes: Few inches
- Everything on one chip: micro-everything
 - processor, transceiver, battery, sensors, memory, bus
 - MICA: 4 MHz, 40 Kbps, 4 KB SRAM / 512 KB Serial Flash, lasts 7 days at full blast on 2 x AA batteries



Examples



Spec, 3/03

- 4 KB RAM
- 4 MHz clock
- 19.2 Kbps, 40 feet
- Supposedly \$0.30

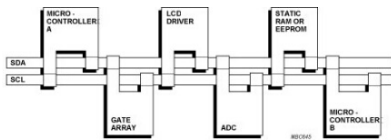


MICA: Current State of the Art (xbow)
Similar i-motes by Intel

Types of Sensors

- Micro-sensors (MEMS, Materials, Circuits)
 - acceleration, vibration, gyroscope, tilt, magnetic, heat, motion, pressure, temp, light, moisture, humidity, barometric, sound
- Chemical
 - CO, CO₂, radon
- Biological
 - pathogen detectors
- [Actuators too (mirrors, motors, smart surfaces, micro-robots)]

I2C bus – simple technology



- Inter-IC connect
 - e.g., connect sensor to microprocessor
- Simple features
 - Has only 2 wires
 - Bi-directional
 - serial data (SDA) and serial clock (SCL) bus
- Up to 3.4 Mbps
- Developed By Philips

Transmission Medium

- Spec, MICA: Radio Frequency (RF)
 - Broadcast medium, routing is “store and forward”, links are bidirectional
- Smart Dust : smaller size => RF needs high frequency => higher power consumption
 - Optical transmission*: simpler hardware, lower power
 - Directional antennas only, broadcast costly
 - Line of sight required
 - Switching links costly : mechanical antenna movements
 - Passive transmission (reflectors) => wormhole routing
 - Unidirectional links

Berkeley Family of Motes

Mote Type	WeC	René	René 2	Dot	Mica	MicaDot
Microcontroller						
Type	AT90LS8535			ATmega163	ATmega128	
Program memory (KB)	8			16	128	
RAM (KB)	0.5			1	4	
Nonvolatile storage						
Chip		24LC256			AT45DB041B	
Connection type		I ² C			SPI	
Size (KB)		32			512	
Default power source						
Type	Lithium	Alkaline	Alkaline	Lithium	Alkaline	Lithium
Size	CR2450	2 x AA	2 x AA	CR2032	2 x AA	3B45
Capacity (mAh)	575	2850	2850	225	2850	1000
Communication						
Radio				TR1000		CC1000
Radio speed (kbps)	10	10	10	10	40	38.4
Modulation type				OOK		ASK
						FSK

Summary: Sensor Node

- Small Size : few mm to a few inches
- Limited processing and communication
 - MHz clock, MB flash, KB RAM, 100's Kbps (wireless) bandwidth
- Limited power (MICA: 7-10 days at full blast)
- Failure prone nodes and links (due to deployment, fab, wireless medium, etc.)
- But easy to manufacture and deploy in large numbers
- Need to offset this with scalable and fault-tolerant OS's and protocols

Sensor-node Operating System

Issues

- Size of code and run-time memory footprint
 - Embedded System OS's inapplicable: need hundreds of KB ROM
- Workload characteristics
 - Continuous ? Bursty ?
- Application diversity
 - Reuse sensor nodes
- Tasks and processes
 - Scheduling
 - Hard and soft real-time
- Power consumption
- Communication

TinyOS design point

- Bursty dataflow-driven computations
 - Multiple data streams => concurrency-intensive
 - Real-time computations (hard and soft)
 - Power conservation
 - Size
 - Accommodate diverse set of applications
- TinyOS:
- Event-driven execution (*reactive* mote)
 - Modular structure (components) and clean interfaces

Programming TinyOS

- Use a variant of C called NesC
- NesC defines *components*
- A component is either
 - A *module* specifying a set of methods and internal storage (~like a Java static class)

A module corresponds to either a hardware element on the chip (e.g., the clock or the LED), or to a user-defined software module

Modules implement and use *interfaces*

 - Or a *configuration*, a set of other components *wired* together by specifying the unimplemented methods invocation mappings
- A complete NesC application then consists of one top level configuration

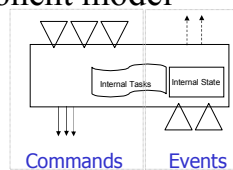
Steps in writing and installing your NesC app

(applies to MICA Mote)

- On your PC
 - Write NesC program
 - Compile to an executable for the mote
 - Plug the mote into the parallel port through a connector board
 - Install the program
- On the mote
 - Turn the mote on, and it's already running your application

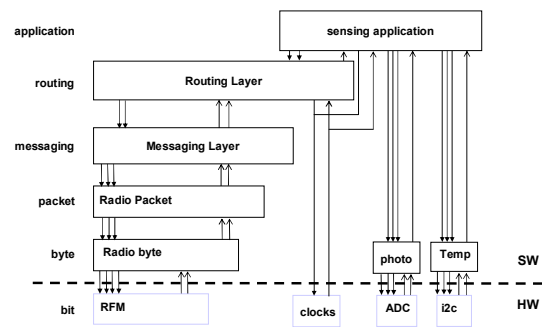
TinyOS component model

- Component specifies:



- Component invocation is event driven, arising from hardware events
- Static allocation avoids run-time overhead
- Scheduling: dynamic, hard (or soft) real-time
- Explicit interfaces accommodate different applications

A Complete TinyOS Application



TinyOS Facts

- Software Footprint 3.4 KB
- Power Consumption on Rene Platform
Transmission Cost: 1 μ J/bit
Inactive State: 5 μ A
Peak Load: 20 mA
- Concurrency support: at peak load CPU is asleep 50% of time
- Events propagate through stack <40 μ S

Energy – a critical resource

- Power saving modes:
 - MICA: active, idle, sleep
- Tremendous variance in energy supply and demand
 - Sources: batteries, solar, vibration, AC
 - Requirements: long term deployment v. short term deployment, bandwidth intensiveness
 - 1 year on 2xAA batteries => 200 uA average current

Energy – a critical resource

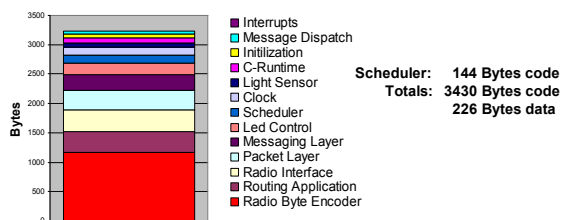
Component	Rate	Startup time	Current consumption
CPU Active	4 MHz	N/A	4.6 mA
CPU Idle	4 MHz	1 us	2.4 mA
CPU Suspend	32 kHz	4 ms	10 uA
Radio Transmit	40 kHz	30 ms	12 mA
Radio Receive	40 kHz	30 ms	3.6 mA
Photo	2000 Hz	10 ms	1.235 mA
I2C Temp	2 Hz	500 ms	0.150 mA
Pressure	10 Hz	500 ms	0.010 mA
Press Temp	10 Hz	500 ms	0.010 mA
Humidity	500 Hz	500 ms	0.775 mA
Thermopile	2000 Hz	200 ms	0.170 mA
Thermistor	2000 Hz	10 ms	0.126 mA

TinyOS: More Performance Numbers

- Byte copy – 8 cycles, 2 microsecond
- Post Event – 10 cycles
- Context Switch – 51 cycles
- Interrupt – h/w: 9 cycles, s/w: 71 cycles

TinyOS: Size

Code size for ad hoc networking application



TinyOS: Summary

Matches both

- **Hardware requirements**
 - power conservation, size
- **Application requirements**
 - diversity (through modularity), event-driven, real time

Discussion

System Robustness

- @ Individual sensor-node OS level:
 - Small, therefore fewer bugs in code
 - TinyOS: efficient network interfaces and power conservation
 - Importance? Failure of a few sensor nodes can be made up by the distributed protocol
- @ Application-level ?
 - Need: Designer to know that sensor-node system is flaky
- @ Level of Protocols?
 - Need for fault-tolerant protocols
 - Nodes can fail due to deployment/fab; communication medium lossy
 - e.g., ad-hoc routing to base station:
 - TinyOS's *Spanning Tree* Routing: simple but will partition on failures
 - Need: denser graph - more robust, but more expensive maintenance
 - Application specific, or generic but tailorable to application ?

Scalability

- @ OS level ?
 - TinyOS:
 - Modularized and generic interfaces admit a variety of applications
 - Correct direction for future technology
 - Growth rates: data > storage > CPU > communication > batteries
 - Move functionality from base station into sensor nodes
 - In sensor nodes, move functionality from s/w to h/w
- @ Application-level ?
 - Need: Applications written with scalability in mind
 - Need: Application-generic scalability strategies/paradigms
- @ Level of protocols?
 - Need: protocols that scale well with a thousand or a million nodes

Etcetera

- Option: ASICs versus generic-sensors
 - Performance vs. applicability vs money
 - Systems for sets of applications with common characteristics
- Event-driven model to the extreme: Asynchronous VLSI
- Need: Self-sufficient sensor networks
 - In-network processing, monitoring, and healing
- Need: Scheduling
 - Across networked nodes
 - Mix of real-time tasks and normal tasks
- Need: Security, and Privacy
- Need: Protocols for anonymous sensor nodes
 - E.g., Directed Diffusion protocol

Summary: Distributed Protocols for Sensor Systems...

- ...should match with both
- **Hardware** (e.g., energy use, small memory footprint, fault-tolerance, scalability)
- **Application requirements** (e.g., generic, scalability, fault-tolerance)
- CS 598 IG has a (limited number of) motes available for projects. Ask.

Other Projects

- Berkeley
 - TOSSIM (+TinyViz)
 - TinyOS simulator (+ visualization GUI)
 - TinyDB
 - Querying a sensor net like a database
 - Maté, Trickle
 - Virtual machine for TinyOS motes, code propagation in sensor networks for automatic reprogramming, like an active network.
 - CITRIS
- Several projects in other universities too
 - UI, UCLA: networked vehicle testbed

Civilian Mote Deployment Examples

- Environmental Observation and Forecasting (EOFS)
- Collecting data from the Great Duck Island
 - See <http://www.greatduckisland.net/index.php>
- Retinal prosthesis chips

Presentations

- Hurry! Not too many slots left
- Please check course website periodically for “News” updates
- Use the newsgroups `class.cs598ig` to search for partners/chat about topics
- If you did not get your favorite session, don’t worry. Do well on your first topic, and you’ll get a second chance on another favorite.
- Your presentation topic need not be related to your project topic

Next Lecture

- Please read (very carefully) papers on “Impossibility of Consensus” and “Time, clocks and the ordering of events”