ECE 299

Romit Roy Choudhury

(Acknowledgment to Indranil Gupta for the slides)
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

- Hardware
- Operating Systems and Protocols
- 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: Berkeley “Motes” or “Smart Dust”

Passive CCR comm.
MEMS/polysilicon

Sensor
MEMS/bulk, surface, ...

Laser diode
III-V process

Analog I/O, DSP, Control
COTS CMOS

Power capacitor
Multi-layer ceramic

Solar cell
CMOS or III-V

Thick film battery
Sol/gel $V_2O_5$

Can you identify the 4 components here?

1-2 mm
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)
- **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
    - Typical ad-hoc routing is store-and-forward
  - Unidirectional links
# Berkeley Family of Motes

<table>
<thead>
<tr>
<th>Mote Type</th>
<th>WeC</th>
<th>René</th>
<th>René 2</th>
<th>Dot</th>
<th>Mica</th>
<th>MicaDot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>AT90LS8535</td>
<td>ATmega163</td>
<td>ATmega128</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program memory (KB)</td>
<td>8</td>
<td>16</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM (KB)</td>
<td>0.5</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonvolatile storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chip</td>
<td>24LC256</td>
<td></td>
<td>AT45DB041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection type</td>
<td>I²C</td>
<td></td>
<td>SPI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size (KB)</td>
<td>32</td>
<td></td>
<td>512</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default power source</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Lithium</td>
<td>Alkaline</td>
<td>Alkaline</td>
<td>Lithium</td>
<td>Alkaline</td>
<td>Lithium</td>
</tr>
<tr>
<td>Size</td>
<td>CR2450</td>
<td>2 x AA</td>
<td>2 x AA</td>
<td>CR2032</td>
<td>2 x AA</td>
<td>3B45</td>
</tr>
<tr>
<td>Capacity (mAh)</td>
<td>575</td>
<td>2850</td>
<td>2850</td>
<td>225</td>
<td>2850</td>
<td>1000</td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radio</td>
<td>TR1000</td>
<td></td>
<td>CC1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radio speed (kbps)</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>40</td>
<td>38.4</td>
</tr>
<tr>
<td>Modulation type</td>
<td>OOK</td>
<td></td>
<td>ASK</td>
<td>FSK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 $\Rightarrow$ 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

Routing Layer

Messaging Layer

Radio Packet

Radio byte

RFM

sensing application

photo

Temp

clocks

ADC

i2c

SW

HW

application

routing

messaging

packet

dbyte

bit
TinyOS Facts

- Software Footprint  3.4 KB
- Power Consumption on Rene Platform
  Transmission Cost: 1 $\mu$J/bit
  Inactive State: 5 $\mu$A
  Peak Load: 20 mA
- Concurrency support: at peak load CPU is asleep 50% of time
- Events propagate through stack $<40$ $\mu$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 intensive
  – 1 year on 2xAA batteries => 200 uA average current
Energy – a critical resource

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

Scheduler: 144 Bytes code
Totals: 3430 Bytes code
226 Bytes data
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 processes
- Need: Security, and Privacy
- Need: Protocols for anonymous sensor nodes
  - eg. 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
• There may be reorganization of some session dates
• 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.
Next

• Next few months – Distributed protocols for sensor networks

• Next Class – Please papers on “Impossibility of Consensus” and “Time, clocks and the ordering of events”