Sensor Networks

David Andersen
Low-Power Computing
Carnegie Mellon University

Sensor Evolution

Integration of sensing, computation, and communication
- Low-power, wireless "sensors" with tiny amounts of CPU memory
- Large federated networks for high-resolution sensing of environment

© 2005 M!W

slide credit: Matt Welsh

The Hardware

- Some common µ-processors
- Atmel ATMega128L
  - 128KB flash, 4K EEPROM, 4K SRAM
  - 0-8Mhz adjustable clock
  - 2.7 - 5.5V
- Unlike desktop procs, very detailed power info available. :)

Improvement upon mica2 mote (better sensors, radio)

- XSM (2004)
- Piezo
- Antenna (Centered)

CPU Under Here
**Typical Sensor Stuff**

- CPU, no ADC
- ADC, no CPU
- Timer, no CPU
- Sensor (analog out)

**Sleepy Time**

- Most modules on sensor board can be shut down (power supply gated)
- The CPU itself - can shut down ADC, internal voltages (10 µA, to give an idea of the power range), watchdog, i/o pins
- Idle, Power-down (ext interrupts wake up)

**ATMega 128L Sleeps**

- Active 8Mhz: 20-24mW
- Idle 8Mhz: ~10mW
- Power-down, no watchdog: ~0.5µW
- Power-down, with w/d: ~30µW (paper)
- 1MHz watchdog - non-negligible.
- Safety beats power in unattended sensor
Other option: TI

- MPS430 - 16bit - ex:
  - 16KB RAM, 128KB Flash
  - <= 18MHz
  - 1.8V-3.6V. 5-8.5mW active @ 8MHz; less if program fits in DRAM.
  - $5 or so. Cool toy.

Telos Mote

- CC2420 radio (2.4 GHz, 802.15.4)
- 100dBm, 100-m range

Several thousand produced. Used by 100s of research groups

Great platform for experimentation (though not particularly small)

- Easy to integrate with sensors & actuators
- T&I (now) admits 18kHz down to 5kHz
- 1.4 mBaud expansion (RFI prone, lasts limited by small life of battery)

#s change, game is the same - but favors more compute/less radio.

<table>
<thead>
<tr>
<th>Power</th>
<th>XSM</th>
<th>Telos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standby</td>
<td>30µW</td>
<td>15µW</td>
</tr>
<tr>
<td>Idle</td>
<td>~10mW</td>
<td>~150µW*</td>
</tr>
<tr>
<td>Active</td>
<td>24mW</td>
<td>5mW</td>
</tr>
<tr>
<td>No radio</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RX</td>
<td>55mW</td>
<td>55mW</td>
</tr>
<tr>
<td>TX</td>
<td>72mW</td>
<td>50mW</td>
</tr>
</tbody>
</table>

* Higher in reality!

Irony

- Moteiv (maker of Telos Motes)
- Acquired by Sentilla
- Sentilla now sells...
  - Datacenter energy consumption monitors and analysis software.~)
  - but no motes. ~(TI, however, sells very cute little dev boards.
And *almost*

- 32 bit platforms almost becoming usable for sensor nets
- iMote2 (Crossbow - mica folks)
- Intel PXA271 - 256K SRAM, 32MB SDRAM (woah), 32MB flash
- 31mA active @ 1MHz no radio, 44mA tx/rx. (at sub 1V?) Niice. High sleep current, but give it a few years.

But regardless...

- Reasonable-but-ambitious goal: 1 year, 1 AAA battery.
- 1500 mWh
- There are 8760 hours in a year.
- Avg draw: 171µW
- oof.

Application-Specific Sleeping

- Must sleep a lot.
- Being useful while sleeping a lot: application-specific schedules. Wake only when needed.
- When is it needed?

Two apps

- "Classical" sensor nets:
  - Sample temp & humidity every 5 minutes
  - Send to base station via neighbors
- "Event" sensor nets:
  - Watch frequently, report seldom
  - Events possibly irregular, outside control
**What draws power?**

- The CPU
- The sensors themselves (they're physical devices...)
- The radio
  - Including relaying/collecting/broadcasting

**B-MAC**

- The LPL mode from the paper
- Wake on timer interrupt
- Startup: Wait for XO to stabilize
- Receive: sample signal energy
- Turn radio off, start analyzing signal strength

**BMAC modes**

**MAC Design goals**

- Low power
- Tiny implementation (4616 bytes in ROM, 277 bytes of RAM)
- But nasty to sender: If check channel every 100ms, then Tx preamble must be 100ms long.
- Assumption: Very infrequent Tx.
- No time sync as in BSD/802.11
“Classical” alternative

- Option 1: Batch the heck out of it;
- use LPL
- Option 2: Schedule a wake-up time for reporting
- Much more BSD-like.
- Which? Depends - how up-to-date do your measurements need to be?

Sensors...

- Are a bit of a PITA to program.
- Mica2: 8-bit RISC-like system
- Telos: 16-bit (HUGE improvement, but still...)
- No memory protection, no conventional OS, processes, scheduler, etc. Not even what a rt-OS like VxWorks gives you

TinyOS

- popular OS platform for motes
- Fairly standard OS challenge:
  - Writing individual modules is OK
  - Making the system coherent is hard.
  - Provides a basic scheduler, interrupt support, etc., plus glue to link components

Paranoid Energy Mgmt

- Go back to that picture about timing in BMAC
- Sample channel;
  - tell ADC to take reading;
  - immediately put radio to sleep again;
  - while it’s in that process, figure out if channel was busy
- And hey - maybe I also use the ADC for reading my light sensor... and... and...