Emerging Wireless Networks
Outline

- Scope
  - Wireless sensor network
  - Lower Power Personal area networks
  - Wireless mesh networks

- Focus
  - Technology overview
  - Representative projects/solutions
Embedded Networked Sensing

- Micro-sensors, on-board processing, and wireless interfaces all feasible at very small scale
  - can monitor phenomena “up close”
- Will enable **spatially and temporally dense** environmental monitoring
- Embedded Networked Sensing reveal previously unobservable phenomena

Seismic Structure response
Interaction between ground motion and structure & foundation response

Emergency Medical Care
Motes attached to patients collect vital signs (pulse ox, heart rate, etc.)
Ambulance system makes triage decisions, relays to EMTs
PDAs carried by EMTs receive vital signs and enter into field report
Correlate with patient records at hospital

Contaminant Transport
Understand contaminant transport and fate in real systems

Ecosystems
Develop in situ observation of species and ecosystem dynamics

Adopted from D. Estrin’s Mobicom’02 tutorial
Enabling Technologies

- Embedded
  - Control system w/ Small form factor
  - Untethered nodes

- Networked
  - Exploit collaborative Sensing, action

- Sensing
  - Tightly coupled to physical world

Adopted from D. Estrin’s Mobicom’02 tutorial
RFID

- Radio frequency identifier
- Active
  - Battery powered
  - Read-write and read only versions available
  - Longer read ranges (25 to 100 feet)
  - Higher tag costs ($20 to $70 per tag)
  - 2D location systems possible
  - Example: toll booths
- Passive
  - Powered by reader
  - Read-write and read only versions available
  - Shorter read ranges (Inches to 20 feet)
  - Lower tag costs (at least $1 per tag)
  - Item ID
  - Example: item management
Sensor

- Passive elements: seismic, acoustic, infrared, strain, barometer, humidity, temperature, etc.
- Passive Arrays: imagers (visible, IR), biochemical
- Active sensors: radar, sonar
  - High energy, in contrast to passive elements
Networked Sensor Node

Rice Gnome
TIMSP430F149 16bit (7.3728 MHz) 60kB flash memory and 2kB RAM
900MHz radio, GPS extension module

MANTIS (U. of Colorado)
CC1000 multi-channel radio
ATMEGA 128 microcontroller

Xbow Mote
MICAz 128KB ATMega128L program flash memory
512K data memory
2.4GHz 802.15.4 Zigbee radio (250 kbps)
H/W AES-128

Intel Mote
Strong ARM core, SRAM, FLASH, BT radio
Sensor Node Energy Roadmap

- Deployed (5W)
- PAC/C Baseline (.5W)
- (50 mW)
- (1mW)

Source: ISI & DARPA PAC/C Program
Networked Sensor Software

- TinyOS: a sensor operating system
  - nesC: even-driven programming model
  - A graph of components
- Maté: virtual machine that allows online reprogramming
- TOSSIM: simulator for TinyOS networks
  - TinyViz
  - Tython
  - PowerTOSSIM

http://www.tinyos.net/
TinyOS Overview

- Concurrency: event-driven architecture
- Modularity:
  - Scheduler + a graph of components compiled into executables
- Efficiency
  - Event/command -- function calls
  - Few context switches
    - FIFO scheduler, non-preemptive
  - No kernel/application boundary
TinyOS Component Model

- Component has:
  - Frame (storage)
  - Tasks (computation)
  - Command and Event Interface

- Each component declares the commands it uses and the events it signals.
- Statically allocated, fixed sized frames
- Tasks run to completion but can be preempted by events
- Non-blocking requests to lower level components
  - Cannot signal events

Image courtesy Jason Hill et al
TinyOS Component Model (cont’d)

- Events
  - Event handlers deal with hardware events (interrupts) directly or indirectly
  - Deposit information into a frame
  - Post tasks
  - Signal higher level events
  - Call lower level commands

- The scheduler puts the processor to sleep when the task queue is empty.
- Peripherals keep operating and can wake up the processor.
An Example of TOS Component

//AM.comp//

TOS_MODULE AM;
ACCEPTS{
    char AM_SEND_MSG(char addr, char type, char* data);
    void AM_POWER(char mode);
    char AM_INIT();
};

SIGNALS{
    char AM_MSG_REC(char type, char* data);
    char AM_MSG_SEND_DONE(char success);
};

HANDLES{
    char AM_TX_PACKET_DONE(char success);
    char AM_RX_PACKET_DONE(char* packet);
};

USES{
    char AM_SUB_TX_PACKET(char* data);
    void AM_SUB_POWER(char mode);
    char AM_SUB_INIT();
};
Internal Component Graph

Application

Ad hoc Routing Application

Active Messages

Radio Packet

Serial Packet

Temp

UART

I2C

Photo

RFM

Clocks

Radio byte

Packet

Byte

Bit

Slide courtesy Jason Hill et al
The Communications Stack

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>* ReliableComm</td>
</tr>
<tr>
<td>GenericComm</td>
</tr>
<tr>
<td>AMStandard</td>
</tr>
<tr>
<td>* RadioCRCpacket</td>
</tr>
<tr>
<td>RFComm</td>
</tr>
<tr>
<td>* ChannelMonC</td>
</tr>
<tr>
<td>* Chipcon</td>
</tr>
<tr>
<td>SecDedEncoding</td>
</tr>
<tr>
<td>SpiByteFifoC</td>
</tr>
<tr>
<td>Radio</td>
</tr>
</tbody>
</table>

Retransmit dropped packets using Acknowledgement
Calculates CRC.
Packet decomposition and reassembly
Sends and receives data in bytes and notifies data arrival

Setting the parameters for CC1000 radio chip

*: newly made or modified from existing network stack
TOSSIM

- Compile TinyOS components into native binary
  - Can experiment directly new application components
  - Fast and can scale to thousands of nodes
TinyViz

- A visualization tool
PowerTOSSIM

- Simulates the energy consumption of each node in a sensor network
  - Benchmarking power consumption of different modules
  - Output power state transition during simulation
  - Posterior analysis of CPU cycles
- Add module for tracking power state
- Minor modifications to other modules to report transitions
- Decouple trace gathering from processing
CPU Cycle Analysis

- TOSSIM compiles application code into a native PC binary
- Therefore, difficult to determine how many CPU cycles used on the mote
- Could simulate at the AVR instruction level
  - Very slow
- Instead, record runtime basic block execution counts, map basic blocks to cycles used on AVR.
  - This has low overhead
Cycle Count Estimation

**App Code**

```c
if(x>0) {
    t = x+42;
    v = t / pi;
} else {
    v = -1;
}
```

**Transformed Code**

```c
bb[mote][1]++;
if(x>0) {
    bb[mote][2]++;
    t = x+42;
    v = t / pi;
} else {
    bb[mote][3]++;
    v = -1;
}
```
Cycle Count Estimation (2)

App Code

```c
if(x>0) {
  t = x+42;
  v = t / pi;
} else {
  v = -1;
}
```

Compile

Mote Binary

```c
if(x>0) {
  t = x+42;
  v = t / pi;
} else {
  v = -1;
}
```

Analyze Mica2 assembly code:

- Compute number of CPU cycles executed for each basic block

Disassemble and analyze

<table>
<thead>
<tr>
<th>Basic Block</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Cycle Count Estimation

Potential inaccuracies:
- Need accurate mapping from C basic blocks to binary
- Some low level components have no mapping

In many cases, active CPU cycles very small
- Negligible effect on total power
Sample Sensor Projects

- Caveats: the following slides are mostly taken from the corresponding authors’ presentation

- Acoustic localization (UIUC)
- CotsBots (Berkeley)
- Great Duck Island (Berkeley)
- CodeBlue (Harvard)
Localizing Acoustic Events

- Intrusion detection
- Tracking of objects
Lightning Protocol

- Proximity-based localization
  - Find the “closest” sensor
  - “loudest” is not necessarily “closest”!
- Based on the observation that electronic-magnetic wave travels faster than acoustic wave ($c = 3\times 10^8 \text{m/s}$, $v = 340 \text{m/s}$)
- Implementation
  - MICA mote
Goals for CotsBots

- Open-source hardware and software platform for distributed robotics
- Small, cheap, off-the-shelf, and modular robots

- Emphasize application design software for large (> 50) robot networks
- Reduce startup costs to demonstrate distributed multi-robot algorithms
Building an Application: Beep Diffusion

- Simple algorithm to spread robots apart
- Adjusting microphone gain adjusts distance robots move apart
Building an Application: Beep Diffusion

Implementation

- 5 robots in 10ft x 10ft area
- 3.5 second slot time

- Difficulties with buzzer/tone detector hardware lead to inconsistent beep radius
Habitat Monitoring on Great Duck Island

Questions

- What environmental factors make for a good nest? How much can they vary?
- What are the occupancy patterns during incubation?
- What environmental changes occur in the burrows and their vicinity during the breeding season?
Application architecture

- Sensor Node
  - Patch Network
  - Gateway
- Sensor Patch
- Transit Network
- Base-Remote Link
- Basestation
- Client Data Browsing and Processing
- Internet
- Data Service
GDI 2002 deployment
CodeBlue

- Provide routing, naming, discovery, and security for wireless medical sensors, PDAs, PCs used to monitor and treat patients in a range of medical settings.
  - Integration of medical sensors with low-power wireless networks
  - Wireless ad-hoc routing protocols for critical care; security, robustness, prioritization
  - Hardware architectures for ultra-low-power sensing, computation, and communication
  - Interoperation with hospital information systems; privacy and reliability issues
Motes attached to patients collect vital signs (pulse ox, heart rate, etc.)

Ambulance system makes triage decisions, relays to EMTs

PDAs carried by EMTs receive vital signs and enter into field report

Correlate with patient records at hospital
What we have covered

- Key difference between wireless and wireline communication
  - Wireless link quality is highly variable
  - No physical boundary in wireless connectivity
  - Allows mobility

Some quantitative results in propagation model and modulation

How does these characteristics affect the network protocol stack? Or how should the network protocol be (re-)designed differently?

- MAC protocols, IEEE 802.11
- Routing, Mobile IP
- Modification to TCP
- Security
- Power

Enabling new network architecture, new services, new applications

- Multi-hop wireless networks → sensornet, wireless mesh networks etc.
  - Routing protocol
  - Mobility model
  - Location aware applications & mobile services

Educational goals:
- Familiar with network simulation tool, tcl, C++ programming
- Get some hand-on experience
- Learn how to do “experiments”, analyze results, and draw observations
- Learn how to present technical ideas in writing & verbally
- And think critically!