

Wireless Sensor Network

Acknowledgement:

Slides are from D. Estrin et al. Tutorial of Wireless Sensor Networks
At Mobicom 2002. <http://nesl.ee.ucla.edu/tutorials/mobicom02/>

Outline

- Introduction
 - Motivating applications
 - Enabling technologies
 - Unique constraints
 - Application and architecture taxonomy
- Sensor Node Hardware and Software Platforms
- Programming Tools

Embedded Networked Sensing Potential

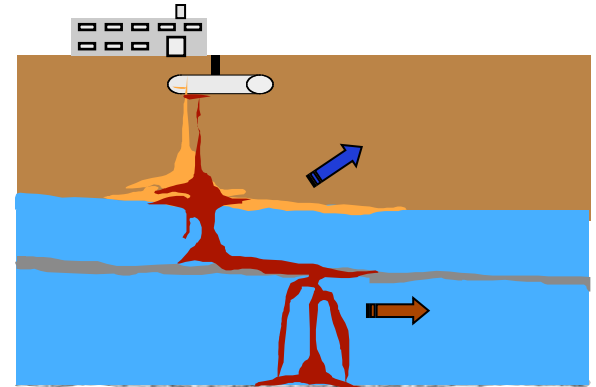


Seismic Structure
response

Marine
Microorganisms

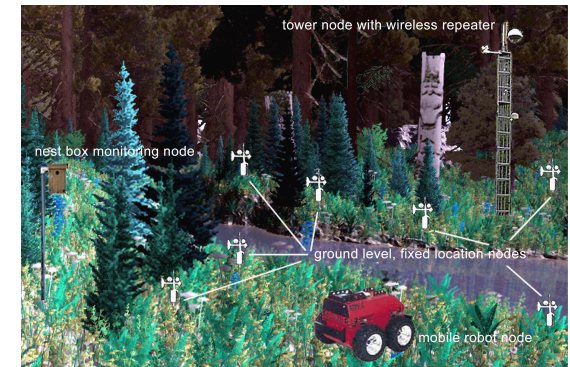


- Micro-sensors, on-board processing, and wireless interfaces all feasible at very small scale
 - can monitor phenomena “up close”
- Will enable **spatially and temporally dense** environmental monitoring
- **Embedded Networked Sensing will reveal previously unobservable phenomena**



Contaminant
Transport

Ecosystems,
Biocomplexity



App#1: Seismic

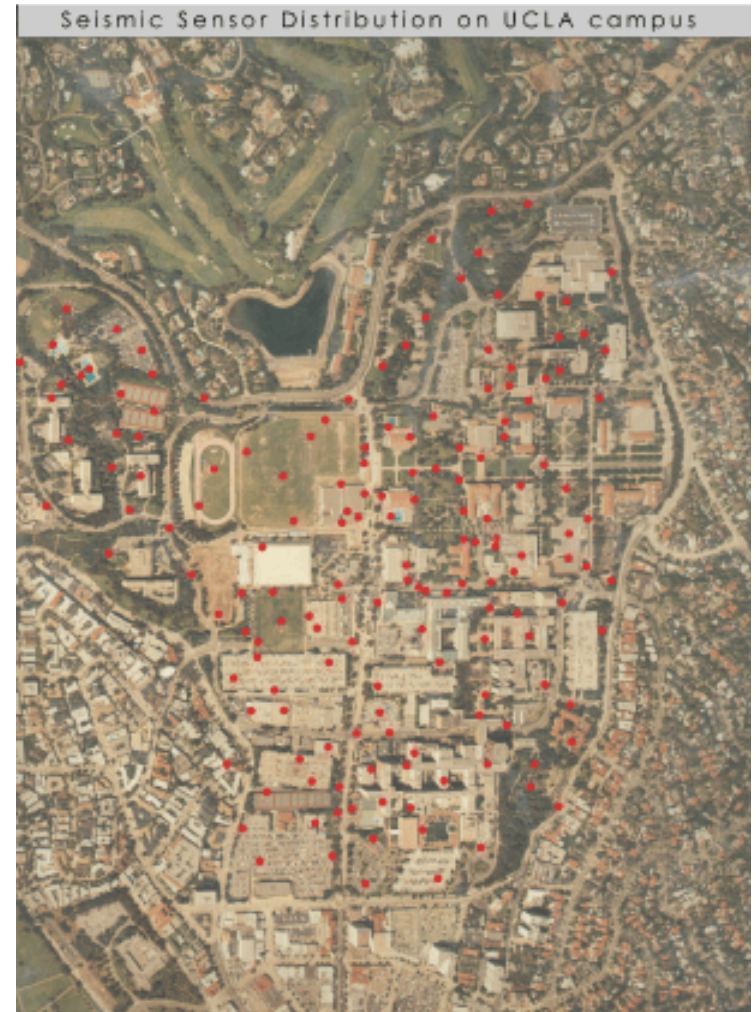
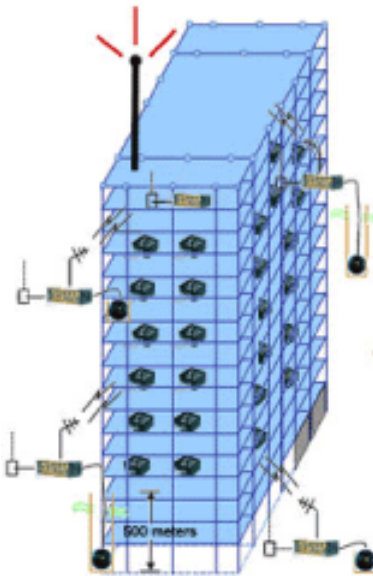
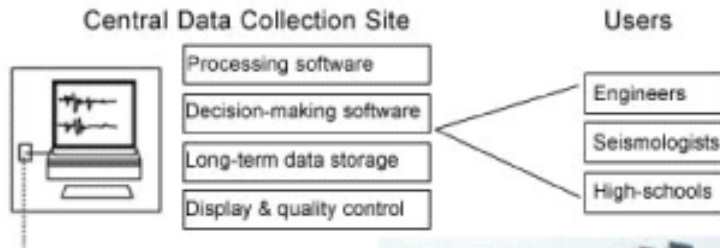


- Interaction between ground motions and structure/foundation response not well understood.
 - **Current seismic networks not spatially dense enough** to monitor structure deformation in response to ground motion, to sample wavefield without spatial aliasing.
- Science
 - Understand response of buildings and underlying soil to ground shaking
 - Develop models to predict structure response for earthquake scenarios.
- Technology/Applications
 - Identification of seismic events that cause significant structure shaking.
 - Local, at-node processing of waveforms.
 - Dense structure monitoring systems.

ENS will provide field data at sufficient densities to develop predictive models of structure, foundation, soil response.

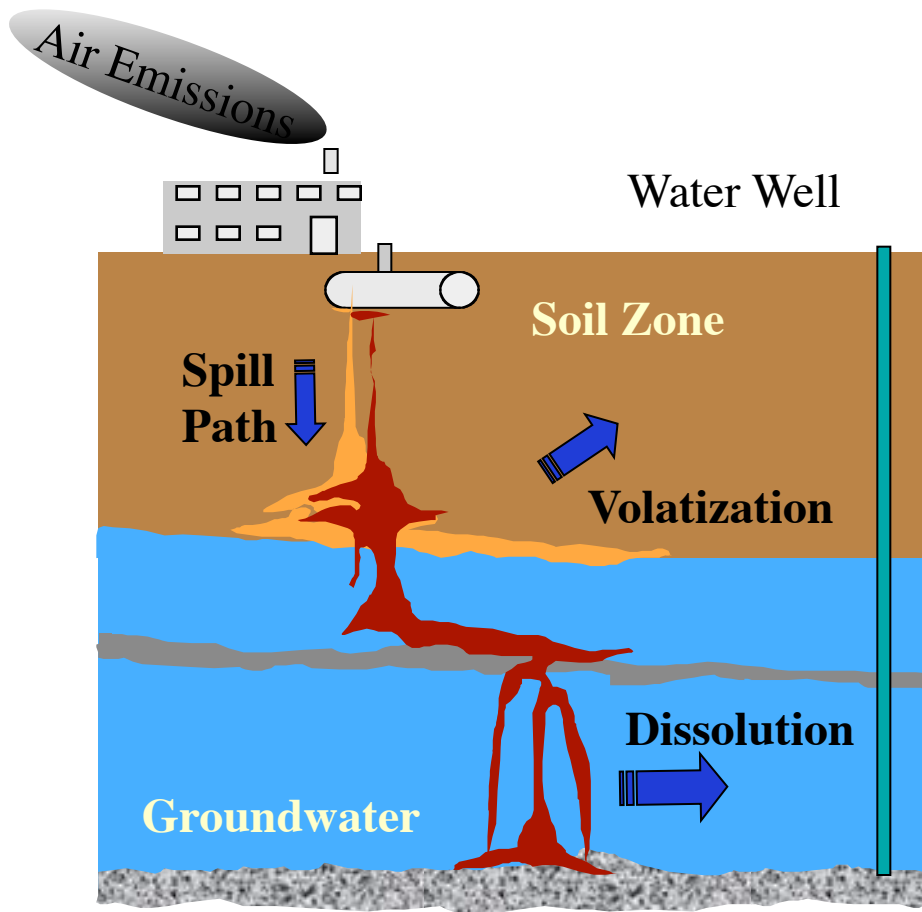
Field Experiment

- 38 strong-motion seismometers in 17-story steel-frame Factor Building.
- 100 free-field seismometers in UCLA campus ground at 100-m spacing



1 km

App#2: Contaminant Transport



- Science
 - Understand intermedia contaminant transport and fate in real systems.
 - Identify risky situations before they become exposures. Subterranean deployment.
- Multiple modalities (e.g., pH, redox conditions, etc.)
- Micro sizes for some applications (e.g., pesticide transport in plant roots).
- Tracking contaminant “fronts”.
- At-node interpretation of potential for risk (in field deployment).

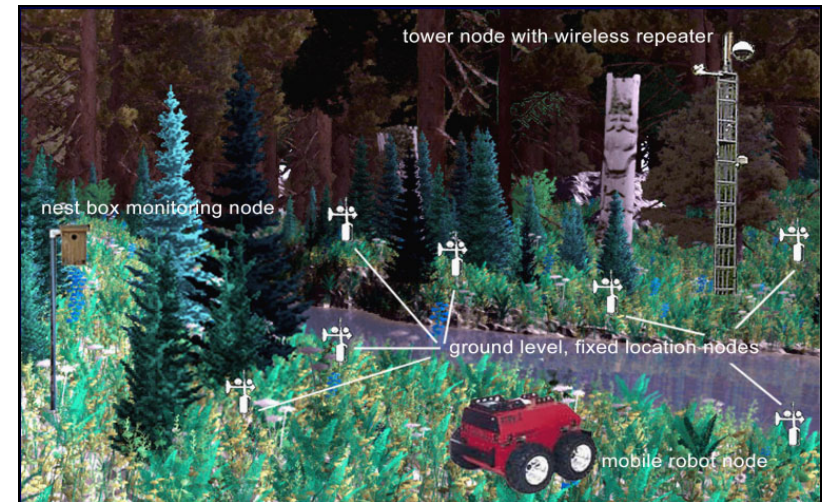
App#3: Ecosystem Monitoring

Science

- Understand response of wild populations (plants and animals) to habitats over time.
- Develop in situ observation of species and ecosystem dynamics.

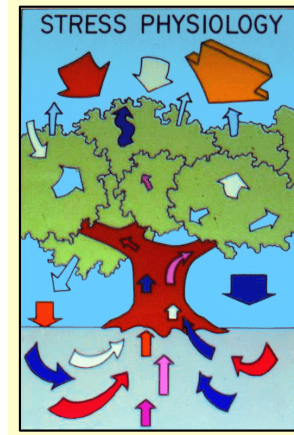
Techniques

- **Data acquisition of physical and chemical properties, at various spatial and temporal scales, appropriate to the ecosystem, species and habitat.**
- Automatic identification of organisms (current techniques involve close-range human observation).
- Measurements over long period of time, taken *in-situ*.
- Harsh environments with extremes in temperature, moisture, obstructions, ...



Field Experiments

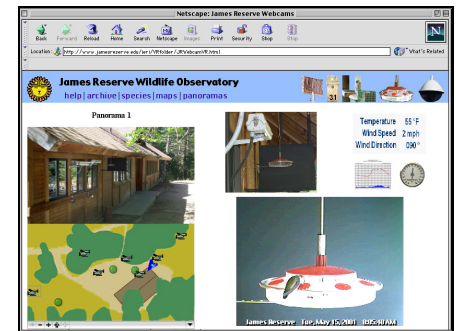
- **Monitoring ecosystem processes**
 - Imaging, ecophysiology, and environmental sensors
 - **Study vegetation response to climatic trends and diseases.**
- **Species Monitoring**
 - Visual identification, tracking, and population measurement of birds and other vertebrates
 - Acoustical sensing for identification, spatial position, population estimation.
- **Education outreach**
 - Bird studies by High School Science classes (New Roads and Buckley Schools).



Vegetation change detection

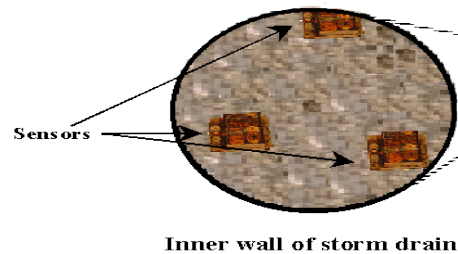
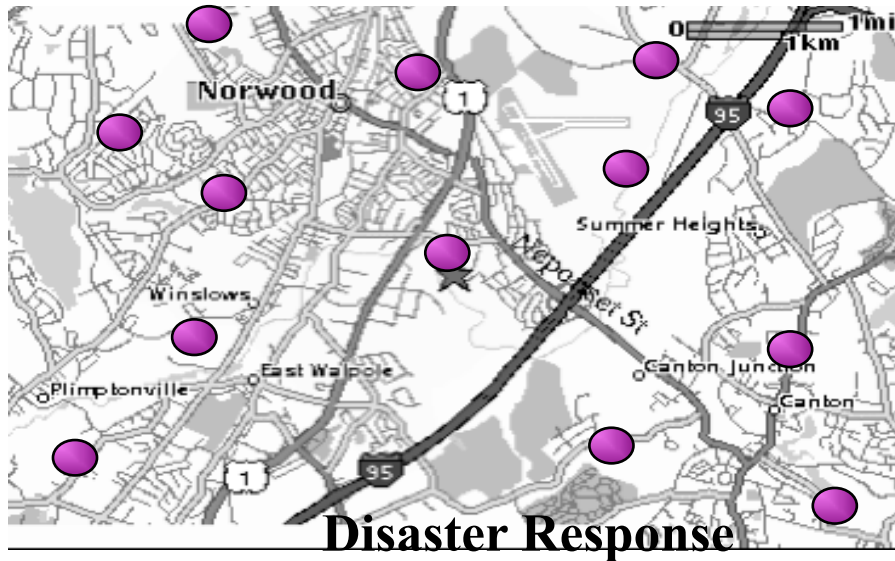


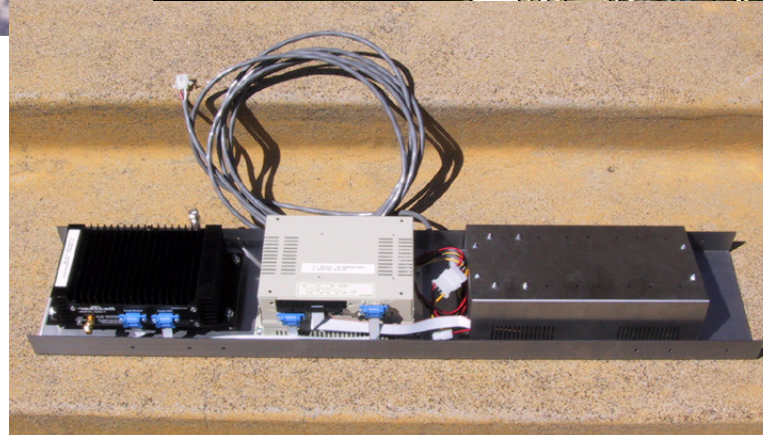
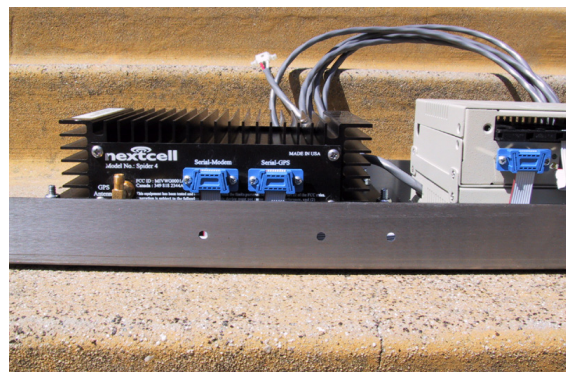
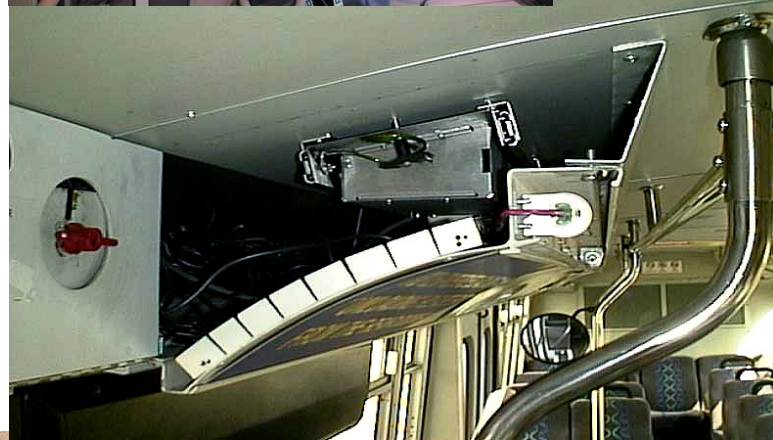
Avian monitoring



Virtual field observations

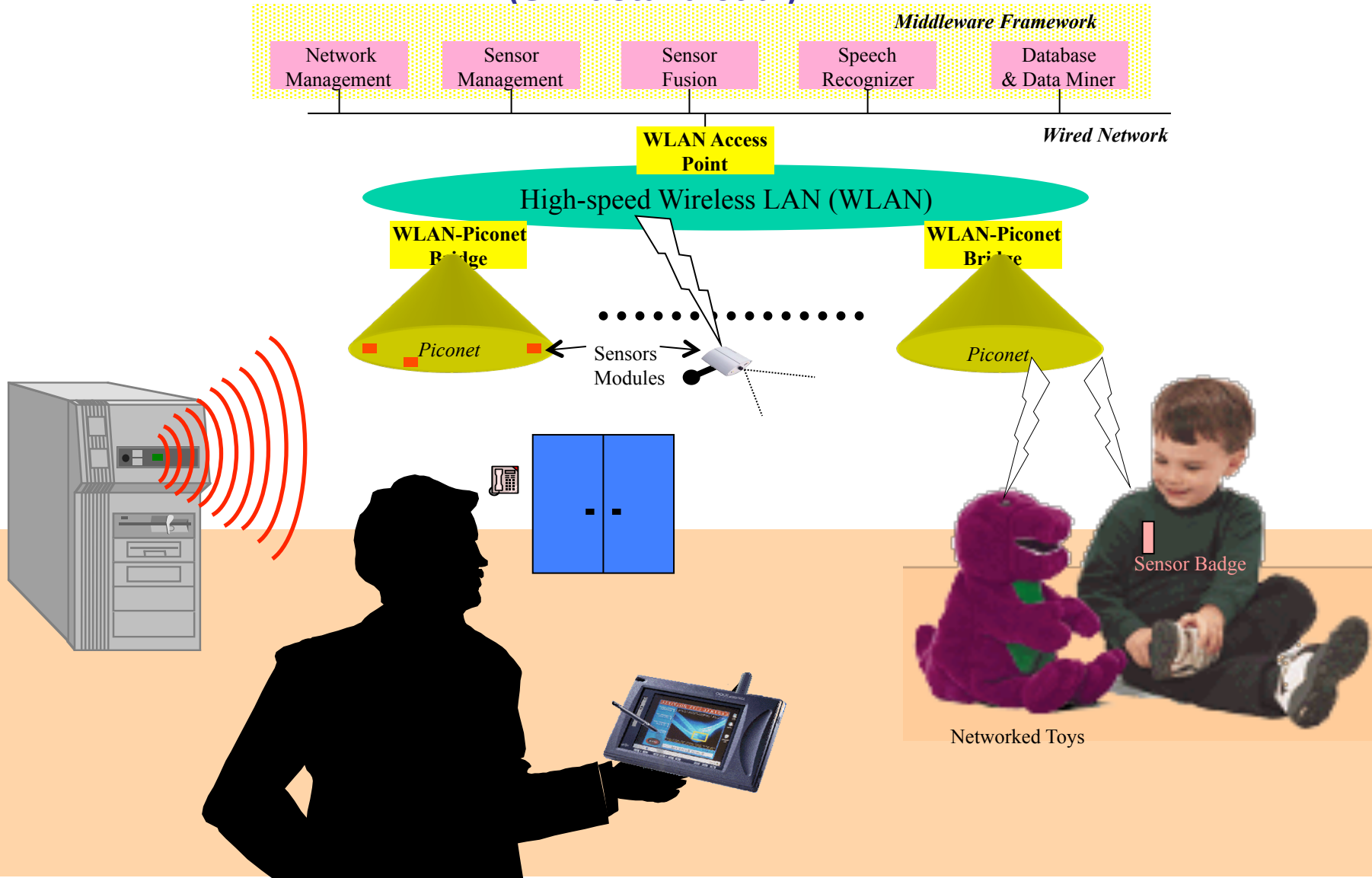
Transportation and Urban Monitoring





Intelligent Transportation Project (Muntz et al.)

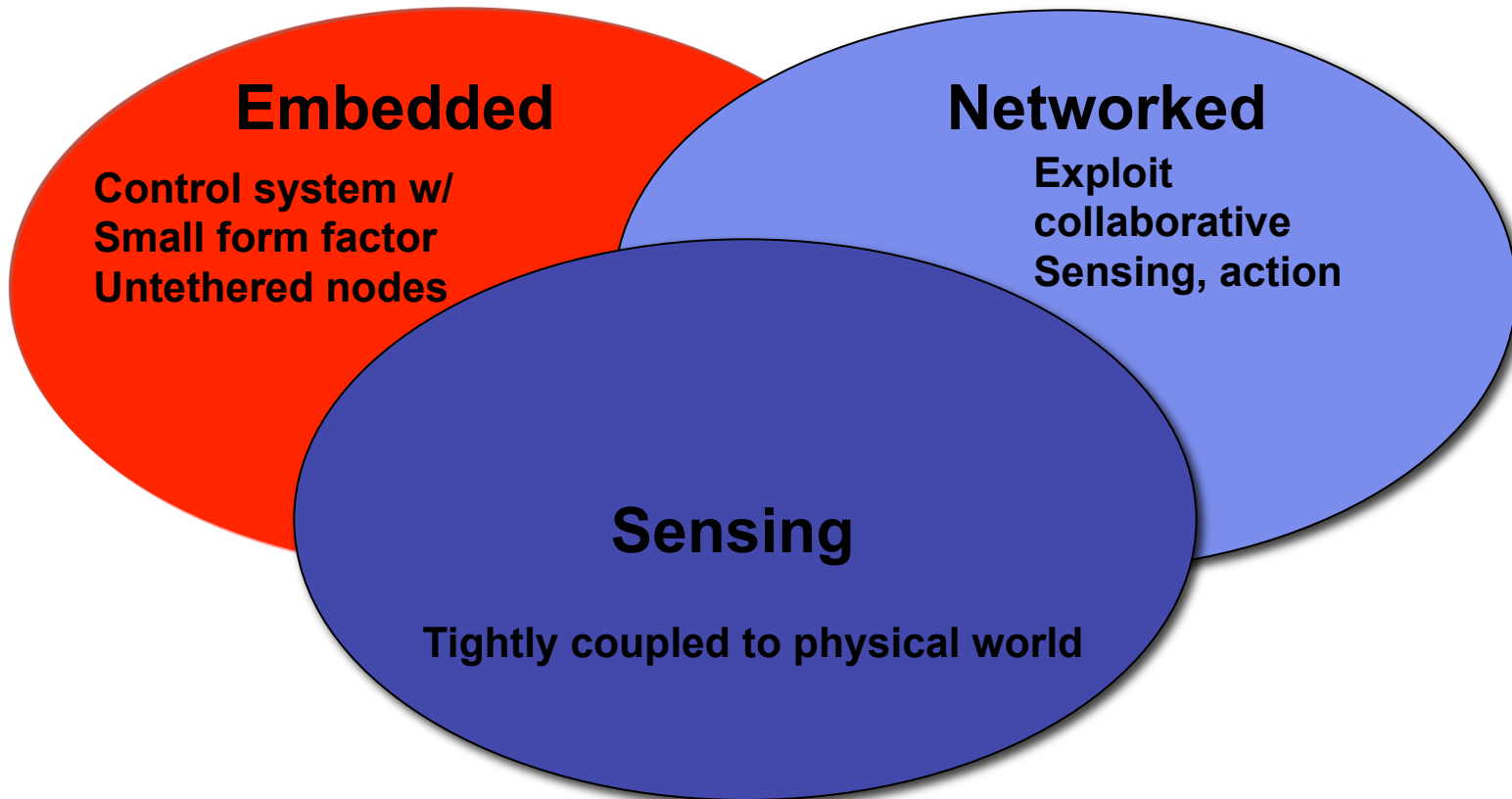
Smart Kindergarten Project: Sensor-based Wireless Networks of Toys for Smart Developmental Problem-solving Environments (Srivastava et al)



Enabling Technologies

Embed numerous distributed devices to monitor and interact with physical world

Network devices to coordinate and perform higher-level tasks



Exploit spatially and temporally dense, in situ, sensing and actuation

Sensors

- Passive elements: seismic, acoustic, infrared, strain, salinity, humidity, temperature, etc.
- Passive Arrays: imagers (visible, IR), biochemical
- Active sensors: radar, sonar
 - High energy, in contrast to passive elements
- Technology trend: use of IC technology for increased robustness, lower cost, smaller size
 - COTS adequate in many of these domains; work remains to be done in biochemical

Some Networked Sensor Node Developments

LWIM III

UCLA, 1996

Geophone, RFM
radio, PIC, star
network



AWAIRS I

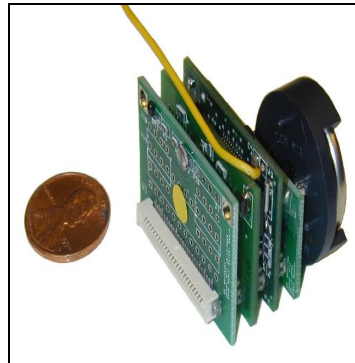
UCLA/RSC 1998

Geophone, DS/SS
Radio, strongARM,
Multi-hop networks



UCB Mote, 2000

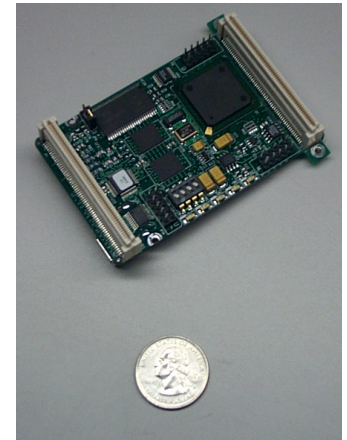
4 Mhz, 4K Ram
512K EEProm,
128K code,
CSMA
half-duplex RFM radio



WINS NG 2.0

Sensoria, 2001

Node development
platform; multi-
sensor, dual radio,
Linux on SH4,
Preprocessor, GPS



Processor

Communication/Computation Technology Projection

	1999 (Bluetooth Technology)	2004
Communication	(150nJ/bit)	(5nJ/bit)
	1.5mW*	50uW
Computation		~ 190 MOPS
		(5pJ/OP)

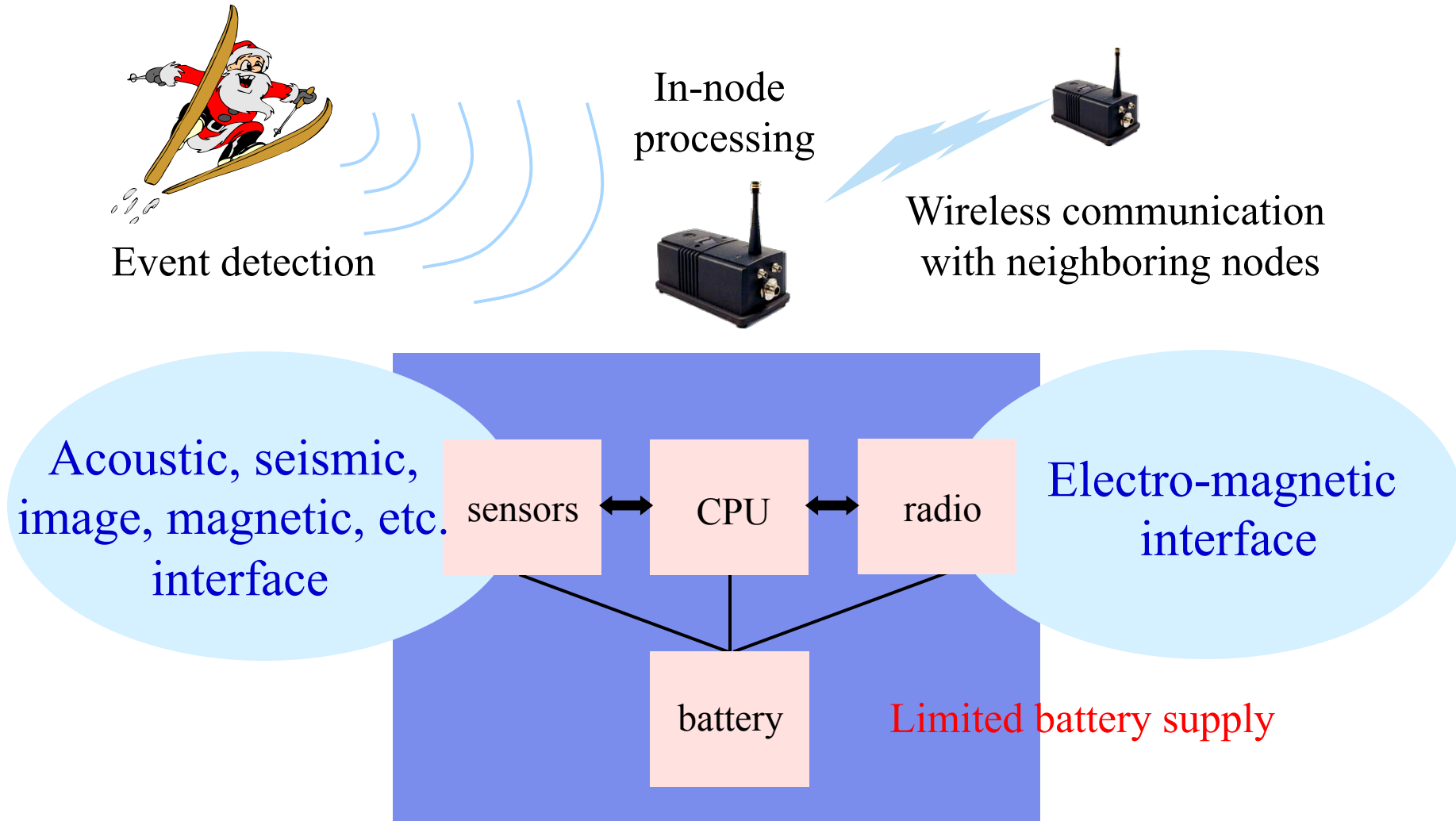
Assume: 10kbit/sec. Radio, 10 m range.

Large cost of communications relative to computation

“The network is the sensor”
(Oakridge National Labs)

**Requires robust distributed systems of
thousands of
physically-embedded, unattended, and often
untethered, devices.**

Sensor Node H/W-S/W Platforms



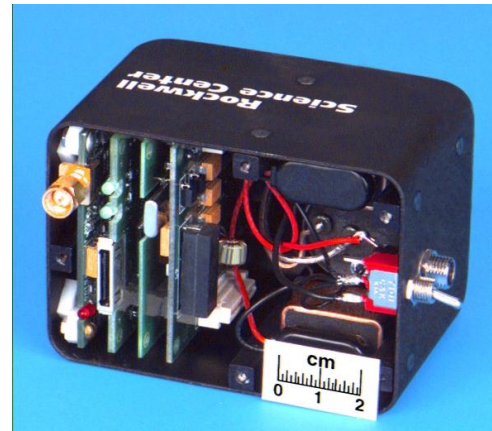
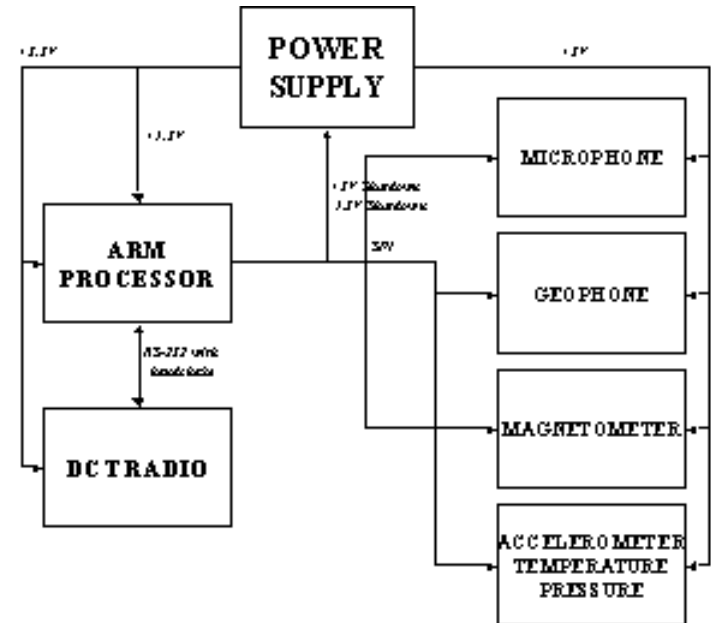
Energy efficiency is the crucial h/w and s/w design criterion

Variety of Real-life Sensor Node Platforms

- RSC WINS & Hydra
 - Sensoria WINS
 - UCLA' s iBadge
 - UCLA' s Medusa MK-II
 - Berkeley' s Motes
 - Berkeley Piconodes
 - MIT' s μ AMPs
 - And many more...
-
- Different points in (cost, power, functionality, form factor) space

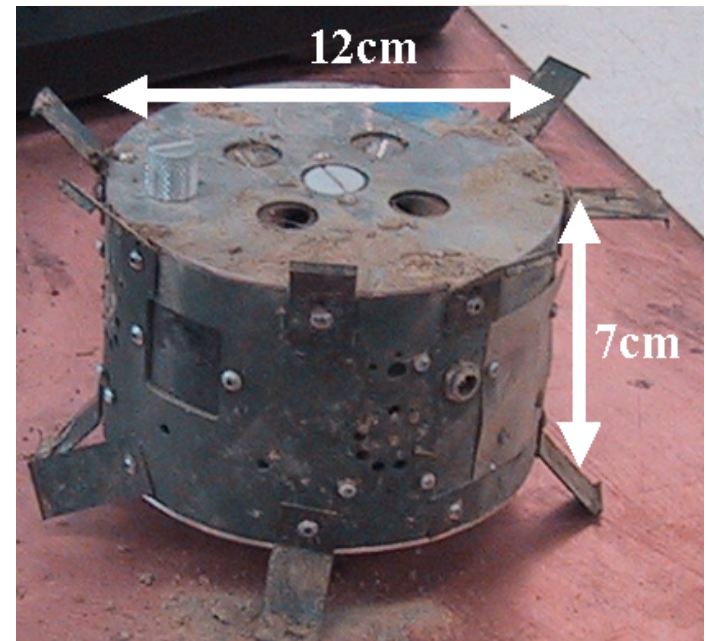
Rockwell WINS & Hydra Nodes

- Consists of 2"x2" boards in a 3.5"x3.5"x3" enclosure
 - StrongARM 1100 processor @ 133 MHz
 - 4MB Flash, 1MB SRAM
 - Various sensors
 - Seismic (geophone)
 - Acoustic
 - magnetometer,
 - accelerometer, temperature, pressure
 - RF communications
 - Connexant's RDSSS9M Radio @ 100 kbps, 1-100 mW, 40 channels
 - eCos RTOS
- Commercial version: Hydra
 - μ C/OS-II
 - TDMA MAC with multihop routing
- <http://wins.rsc.rockwell.com/>

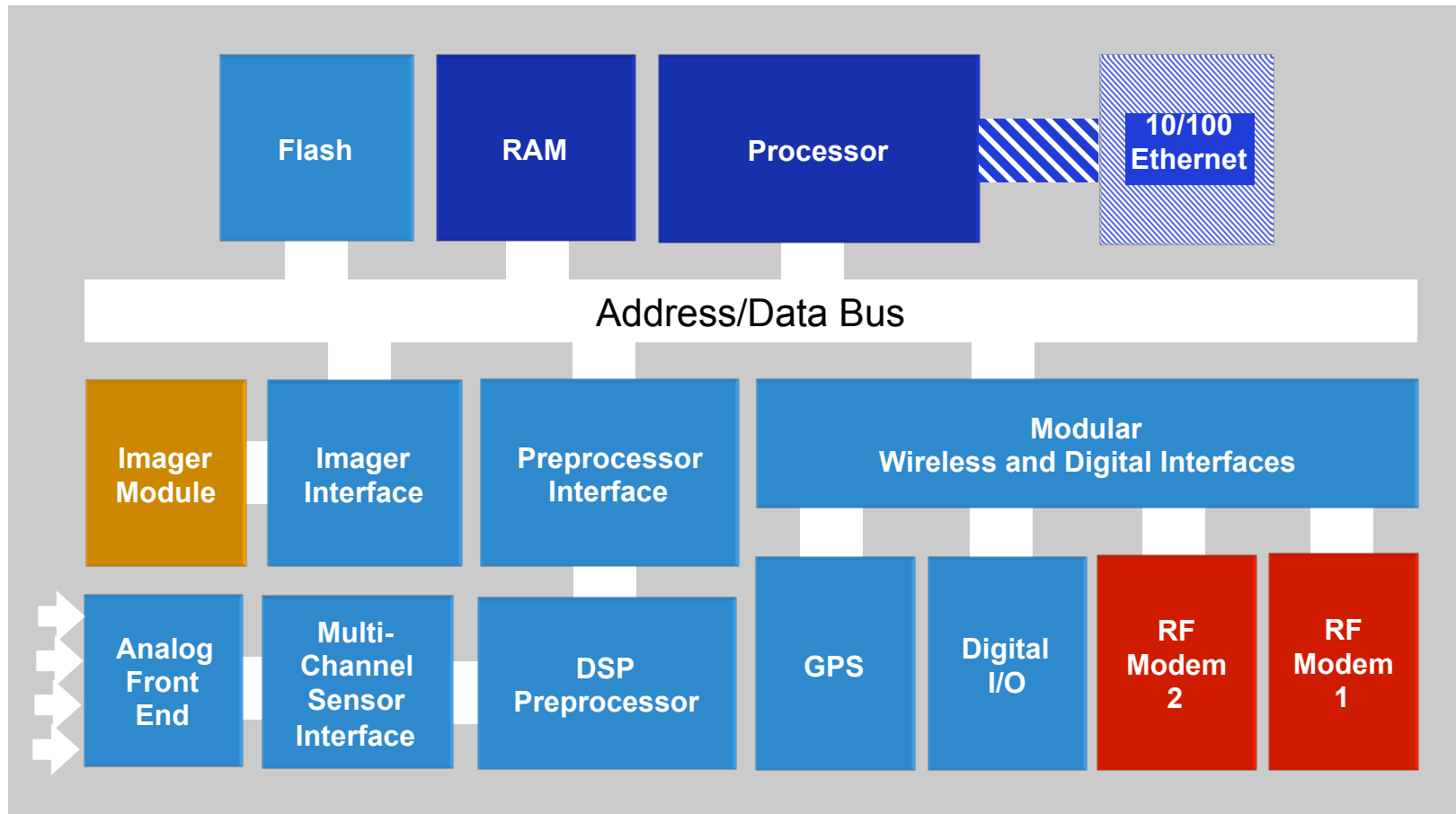


Sensoria WINS NG 2.0, sGate, and WINS Tactical Sensor

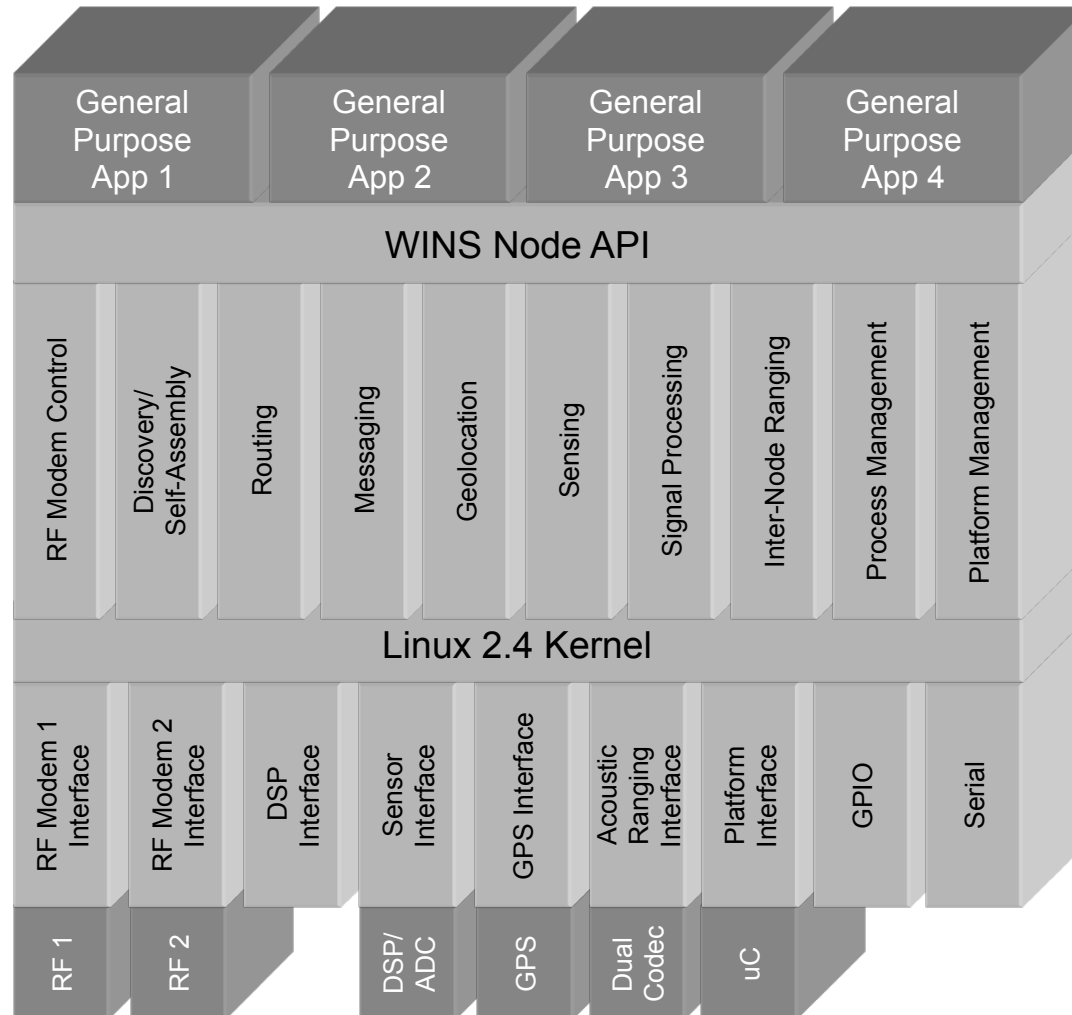
- WINS NG 2.0
 - Development platform used in DARPA SensIT
 - SH-4 processor @ 167 MHz
 - DSP with 4-channel 16-bit ADC
 - GPS
 - imaging
 - dual 2.4 GHz radios
 - Linux 2.4 + Sensoria APIs
 - Commercial version: sGate
- WINS Tactical Sensor Node
 - geo-location by acoustic ranging and angle
 - time synchronization to 5 μ s
 - cooperative distributed event processing



Sensoria Node Hardware Architecture

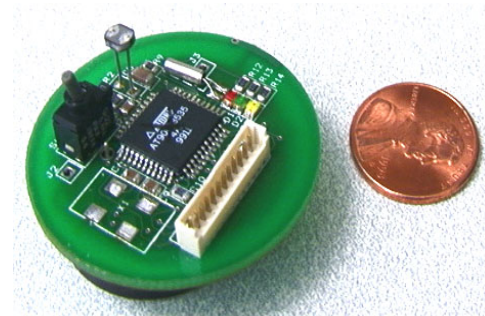
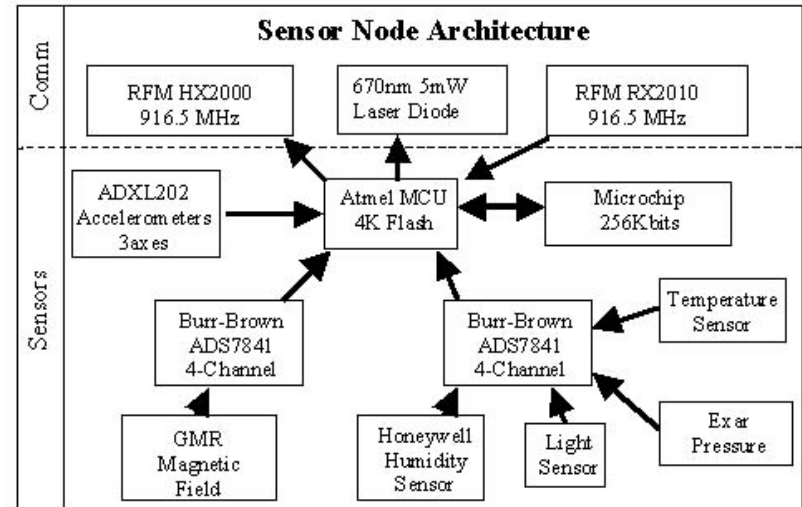


Sensoria Node Software Architecture



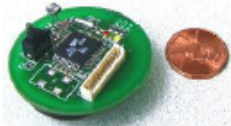



Berkeley Motes

- Devices that incorporate communications, processing, sensors, and batteries into a small package
- Atmel microcontroller with sensors and a communication unit
 - RF transceiver, laser module, or a corner cube reflector
 - temperature, light, humidity, pressure, 3 axis magnetometers, 3 axis accelerometers
- TinyOS



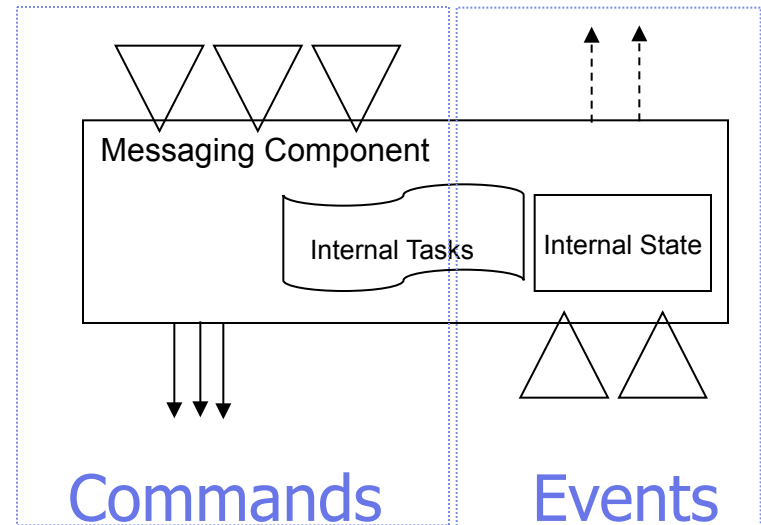
light, temperature,
10 kbps @ 20m

The Mote Family

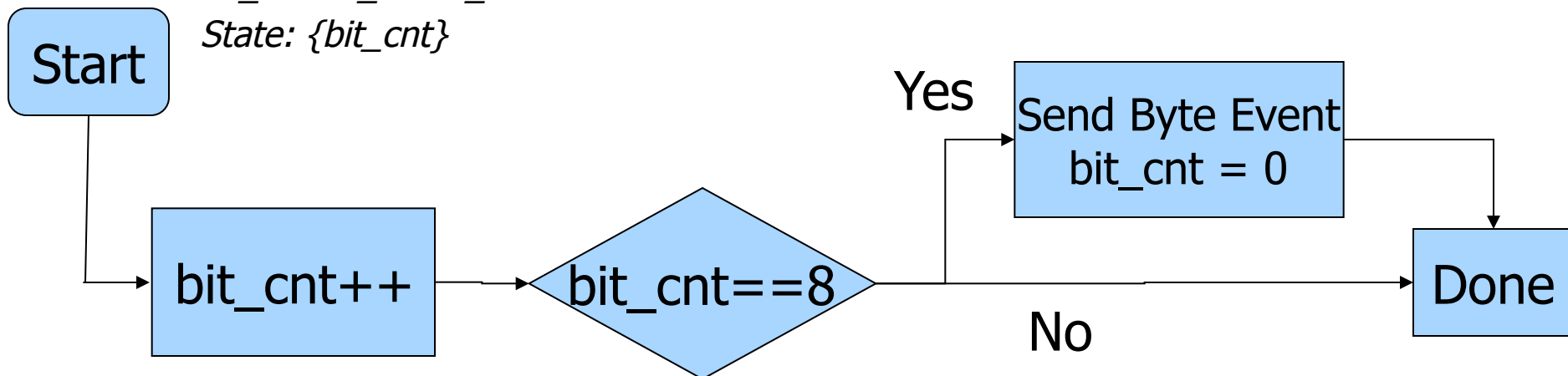
Mote Type	<div>WeC</div>	<div>rene2</div>	<div>rene2</div>	<div>dot</div>	<div>mica</div>
Date	9/99	10/00	6/01	8/01	2/02
Microcontroller					
Type	AT90LS8535		ATMega163		ATMega103
Prog. mem. (KB)	8		16		128
RAM (KB)	0.5		1		4
Nonvolatile storage					
Chip	24LC256				AT45DB041B
Connection type	I2C				SPI
Size (KB)	32				512
Default Power source					
Type	Li	Alk		Li	Alk
Size	CR2450	2xAA		CR2032	2xAA
Capacity (mAh)	575	2850		225	2850
Communication					
Radio	RFM TR1000				
Rate (Kbps)	10	10	10	10	10/40
Modulation type	OOK				OOK/ASK

TinyOS

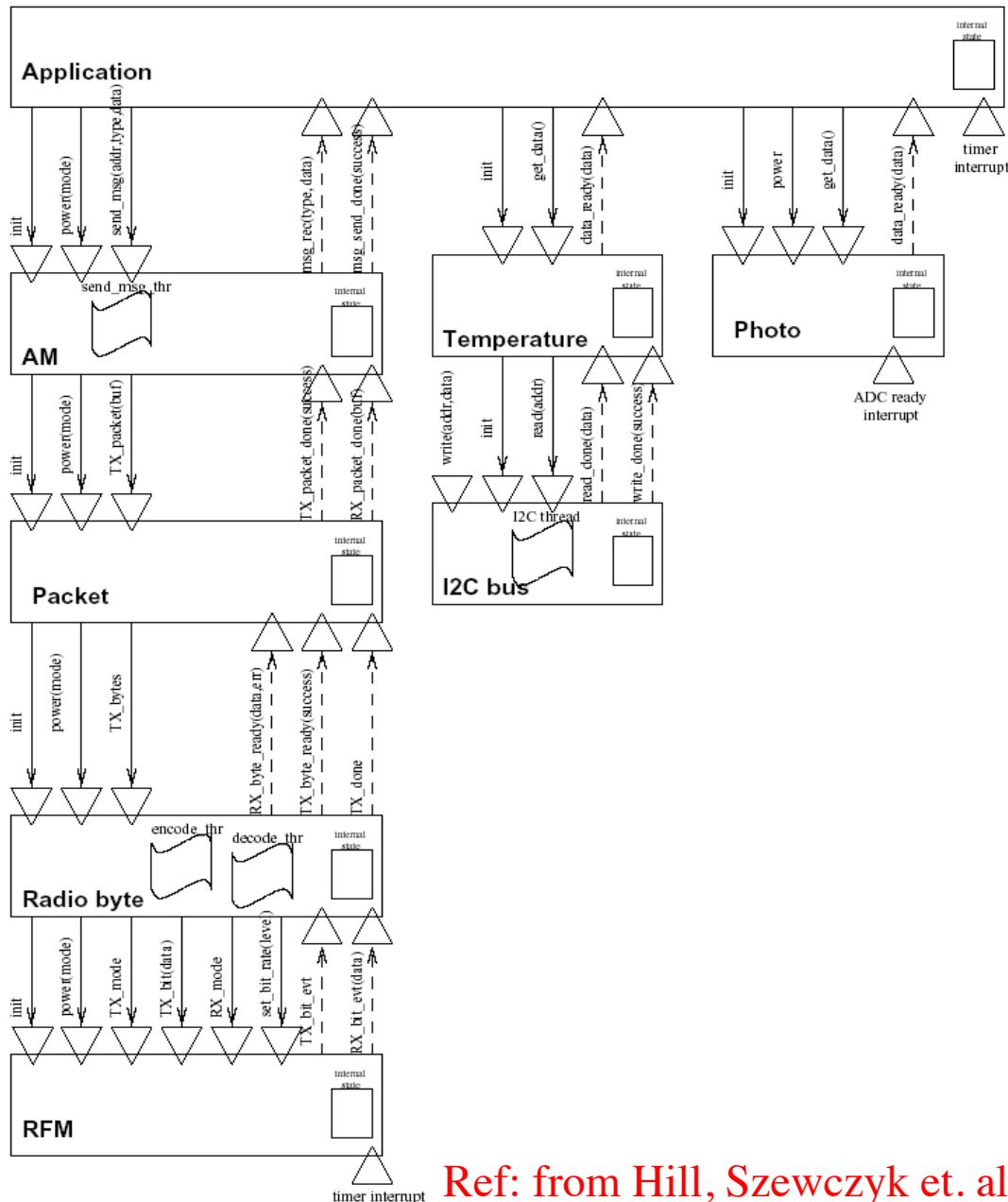
- System composed of concurrent FSM modules
 - Single execution context
- Component model
 - Frame (storage)
 - Commands & event handlers
 - Tasks (computation)
 - Command & Event interface
 - Easy migration across h/w -s/w boundary
- Two level scheduling structure
 - Preemptive scheduling of event handlers
 - Non-preemptive FIFO scheduling of tasks
- Compile time memory allocation
- NestC
- <http://webs.cs.berkeley.edu>



Bit_Arrival_Event_Handler
State: {bit_cnt}



Complete TinyOS Application

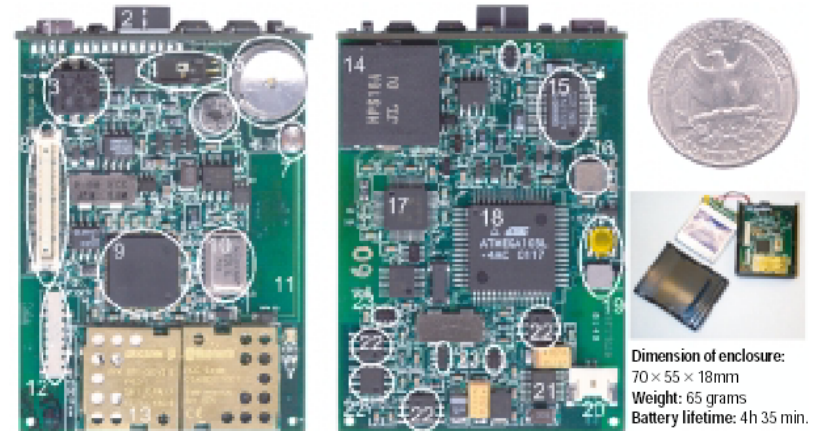
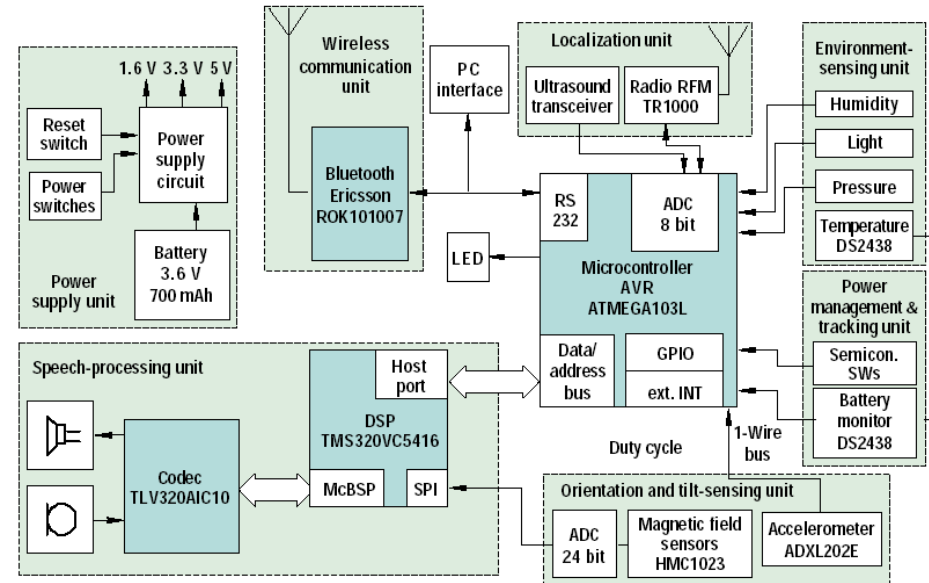


Component Name	Code Size (bytes)	Data Size (bytes)
Multihop router	88	0
AM_dispatch	40	0
AM_temperature	78	32
AM_light	146	8
AM	356	40
Packet	334	40
RADIO_byte	810	8
RFM	310	1
Photo	84	1
Temperature	64	1
UART	196	1
UART_packet	314	40
I2C_bus	198	8
Processor_init	172	30
TinyOS scheduler	178	16
C runtime	82	0
Total	3450	226

Ref: from Hill, Szewczyk et. al., ASPLOS 2000

UCLA iBadge

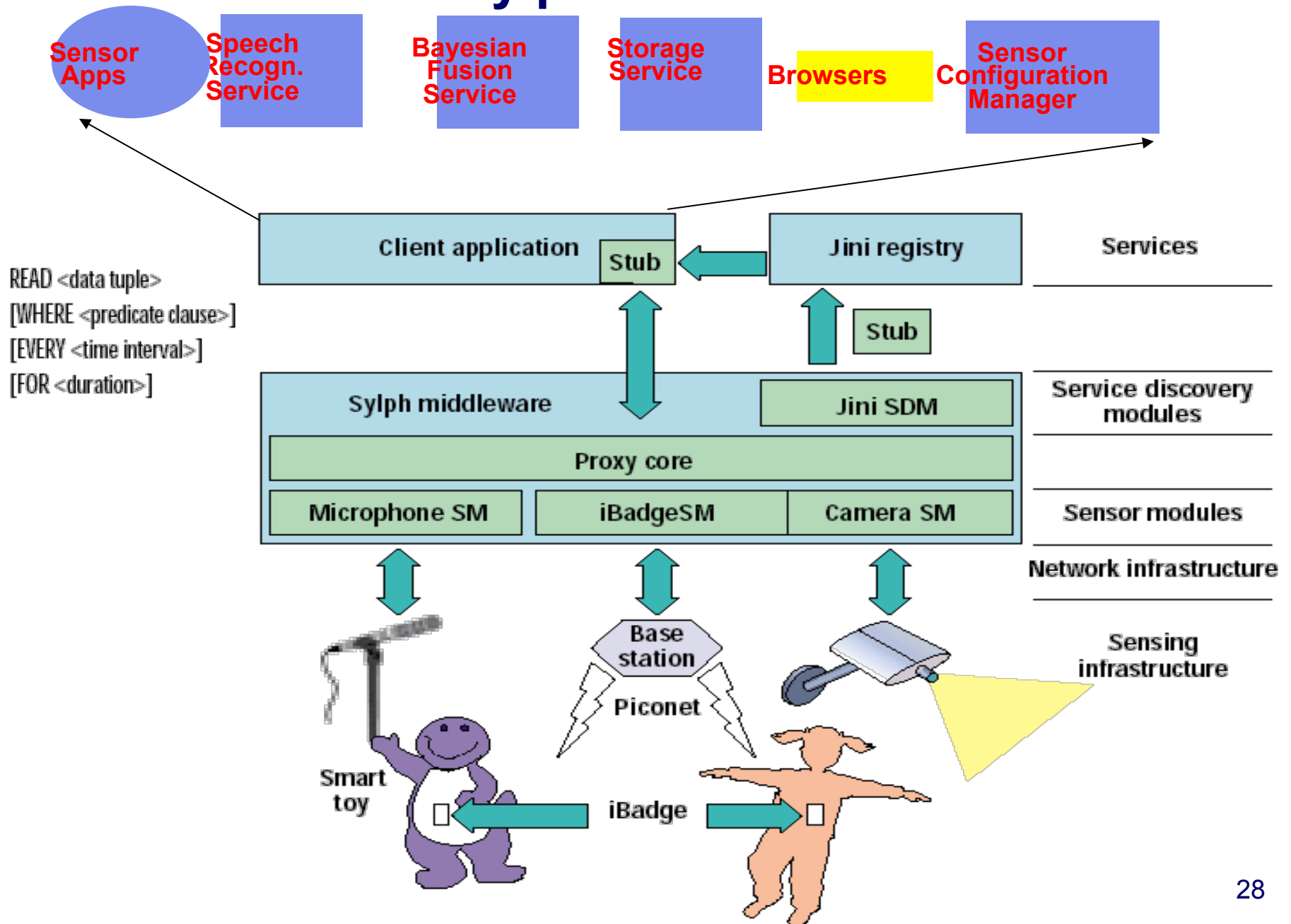
- Wearable Sensor Badge
 - acoustic in/out + DSP
 - temperature, pressure, humidity, magnetometer, accelerometer
 - ultrasound localization
 - orientation via magnetometer and accelerometer
 - bluetooth radio
- Sylph Middleware



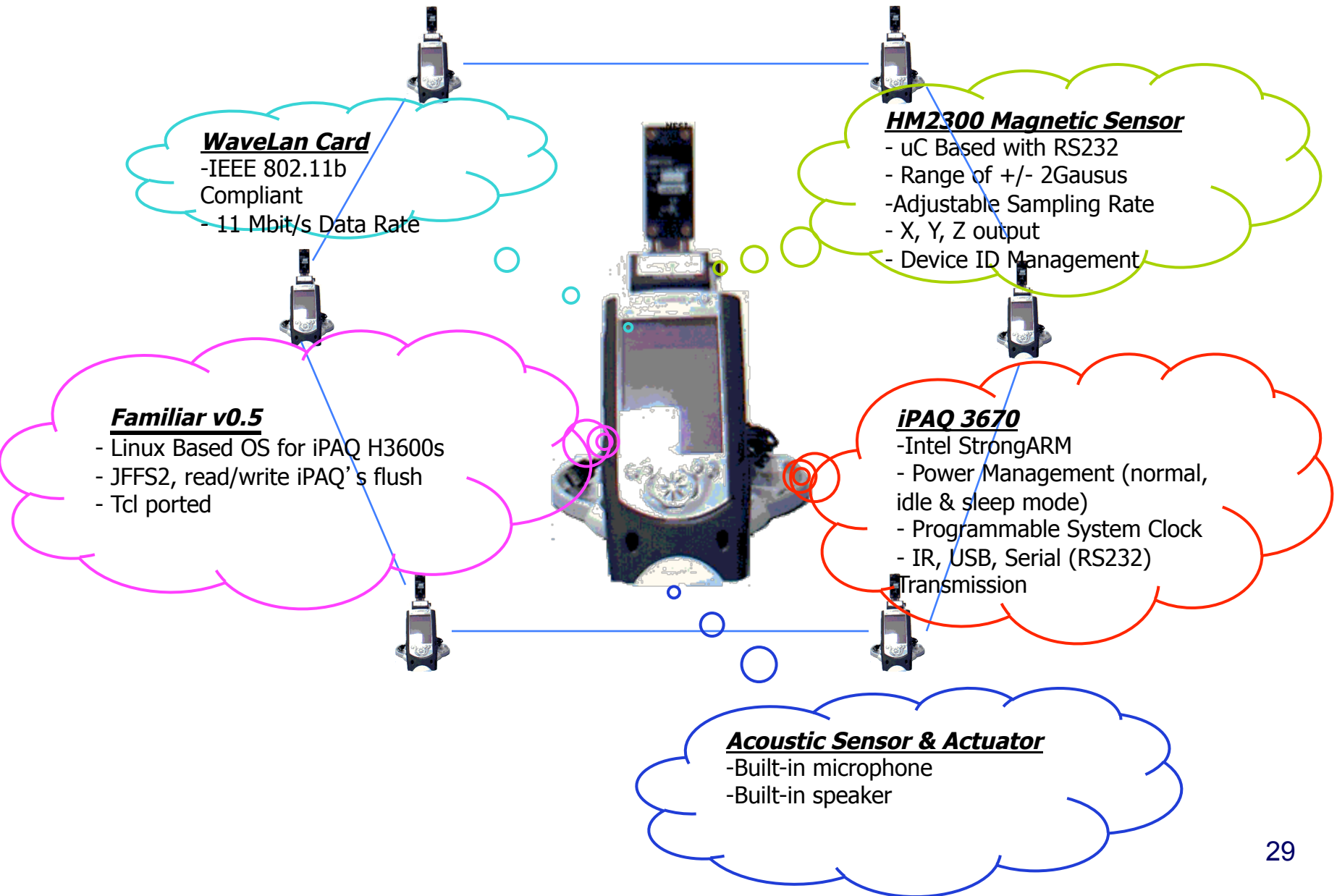
(a) Top
47 × 68 × 7 mm (1.85 × 2.78 × 0.28")

- | | | |
|--------------------------------|----------------------------------|----------------------------------|
| 1. Accelerometer for x, y-axis | 9. DSP | 17. Codec chip |
| 2. Magnetic field sensor | 10. RFM radio (for localization) | 18. Microcontroller |
| 3. Pressure sensor | 11. PCB antenna for RFM radio | 19. Switches (Power, Reset) |
| 4. Humidity sensor | 12. Blue tooth antenna | 20. Battery connector |
| 5. Ultrasound transceiver | 13. Blue tooth module | 21. Power supply |
| 6. Microphone | 14. Loudspeaker | 22. Battery monitors |
| 7. Light sensor | 15. ADC magnetic field sensor | 23. Switches to functional units |
| 8. Connector (SW download) | 16. Accelerometer for x-axis | |

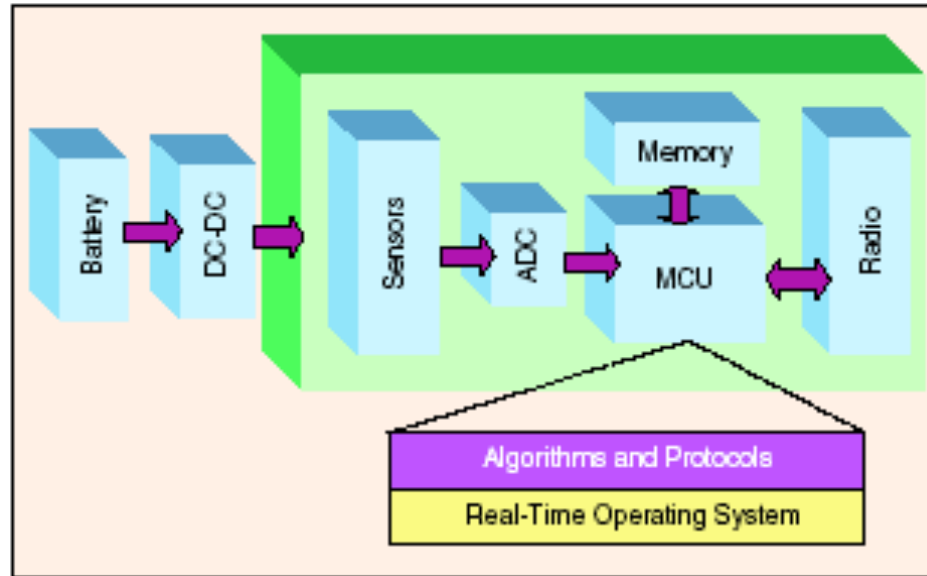
Sylph Middleware



Quick-and-dirty iPaq-based Sensor Node!



Where does the energy go?



- Processing
 - excluding low-level processing for radio, sensors, actuators
- Radio
- Sensors
- Actuators
- Power supply

Processing

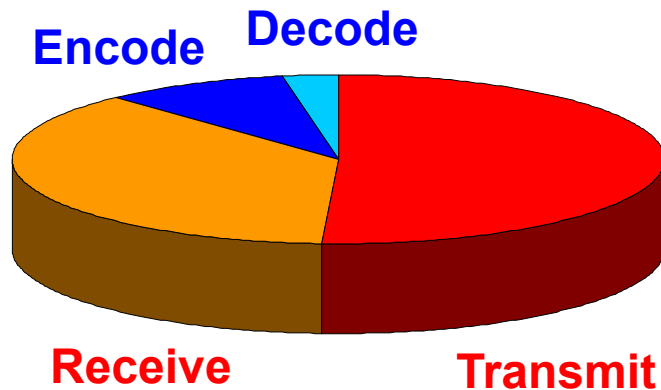
- Common sensor node processors:
 - Atmel AVR, Intel 8051, StrongARM, XScale, ARM Thumb, SH Risc
- Power consumption all over the map, e.g.
 - 16.5 mW for ATmega128L @ 4MHz
 - 75 mW for ARM Thumb @ 40 MHz
- But, don't confuse low-power and energy-efficiency!
 - Example
 - 242 MIPS/W for ATmega128L @ 4MHz (4nJ/Instruction)
 - 480 MIPS/W for ARM Thumb @ 40 MHz (2.1 nJ/Instruction)
 - Other examples:
 - 0.2 nJ/Instruction for Cygnal C8051F300 @ 32KHz, 3.3V
 - 0.35 nJ/Instruction for IBM 405LP @ 152 MHz, 1.0V
 - 0.5 nJ/Instruction for Cygnal C8051F300 @ 25MHz, 3.3V
 - 0.8 nJ/Instruction for TMS320VC5510 @ 200 MHz, 1.5V
 - 1.1 nJ/Instruction for Xscale PXA250 @ 400 MHz, 1.3V
 - 1.3 nJ/Instruction for IBM 405LP @ 380 MHz, 1.8V
 - 1.9 nJ/Instruction for Xscale PXA250 @ 130 MHz, .85V (leakage!)
 - And, the above don't even factor in operand size differences!
- However, need power management to actually exploit energy efficiency
 - Idle and sleep modes, variable voltage and frequency

Radio

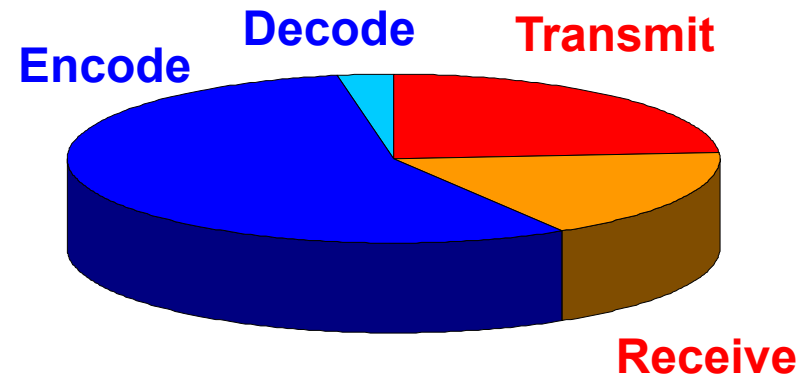
- Energy per bit in radios is a strong function of desired communication performance and choice of modulation
 - Range and BER for given channel condition (noise, multipath and Doppler fading)
- Watch out: different people count energy differently
 - E.g.
 - Mote's RFM radio is only a transceiver, and a lot of low-level processing takes place in the main CPU
 - While, typical 802.11b radios do everything up to MAC and link level encryption in the "radio"
- Transmit, receive, idle, and sleep modes
- Variable modulation, coding
- Currently around 150 nJ/bit for short ranges
- More later...

Computation & Communication

Energy breakdown for voice



Energy breakdown for MPEG



Radio: Lucent WaveLAN at 2 Mbps

Processor: StrongARM SA-1100 at 150 MIPS

- Radios benefit less from technology improvements than processors
- The relative impact of the communication subsystem on the system energy consumption will grow

TinyOS and NesC

Traditional OS

- • Big!
- • Multi-threaded architecture
 - Large number of threads => large memory footprint
- • I/O model
 - Blocking I/O (stop and go): memory per blocked thread
- Kernel and user space separation
- Ample resources and strict boundaries
- In comparison ...

Hardware Constraints

- Power
- Limited memory
- Slow CPU
- Size
- Limited hardware parallelisms
- Short-distance, low-bandwidth radio

Desired OS Properties

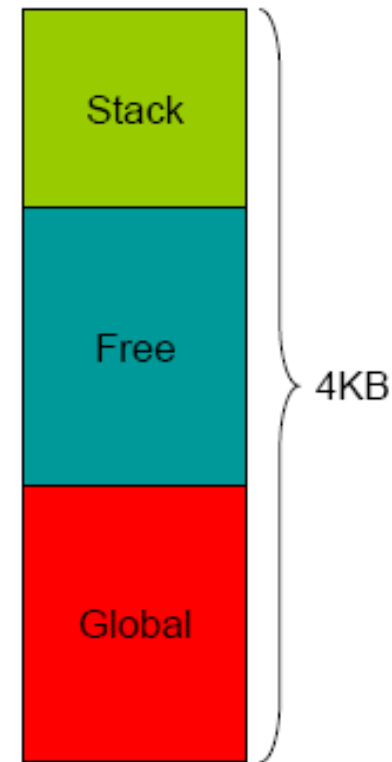
- Small memory footprint
- Efficient in power and computation
- Communication is fundamental
- Real-time
- Support diverse application design

TinyOS Solution

- Support concurrency: event-driven architecture
- Modularity:
 - application = scheduler + graph of components
- Compiled into one executable
- Efficiency: **Get done quickly and sleep**
- Event/command = function calls
 - Fewer context switches: FIFO/non-preemptable scheduling
- No kernel/application boundary

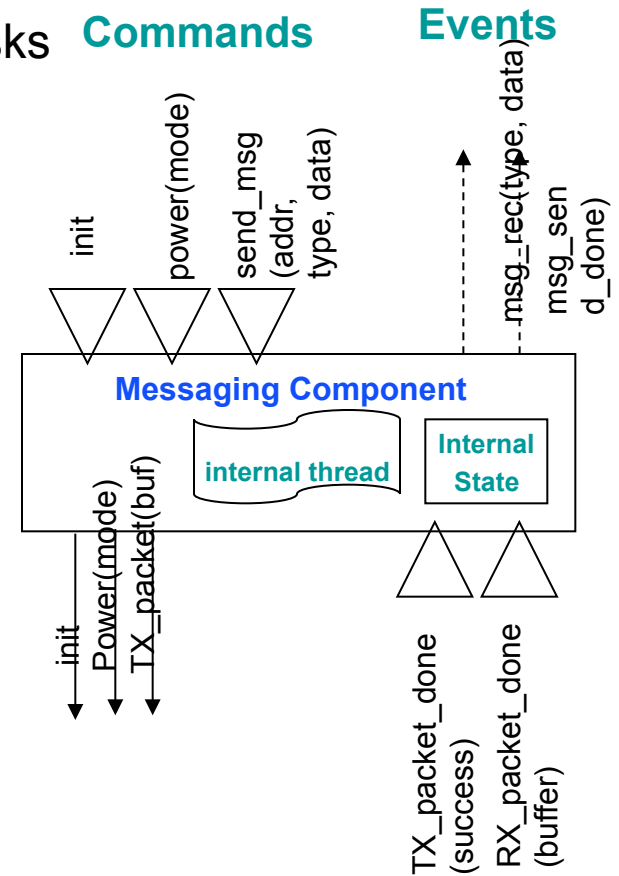
TinyOS Memory Model

- STATIC memory allocation!
 - No heap (malloc)
 - No function pointers
- Global variables
 - Available on a per-frame basis
- Local variables
 - Saved on the stack
 - Declared within a method



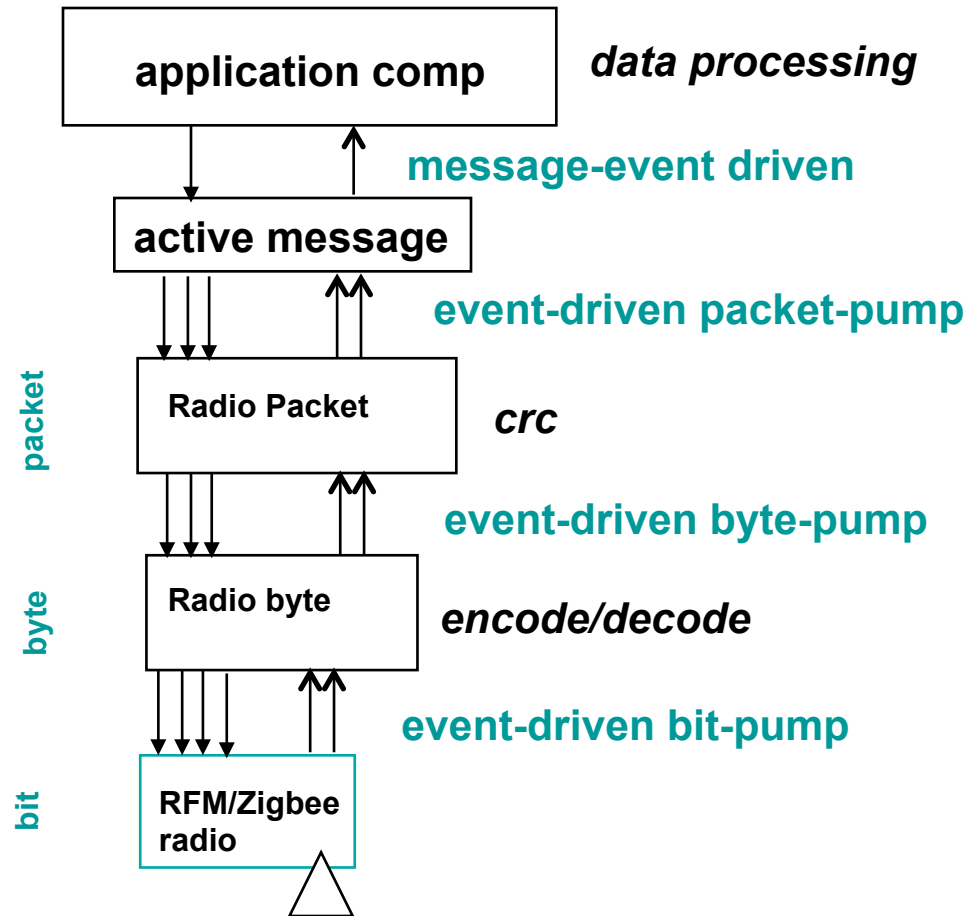
Tiny OS Concepts

- Scheduler + Graph of Components
 - constrained two-level scheduling model: tasks + events
- Component:
 - Commands
 - Event Handlers
 - Frame (storage)
 - Tasks (concurrency)
- Constrained Storage Model
 - frame per component, shared stack, no heap

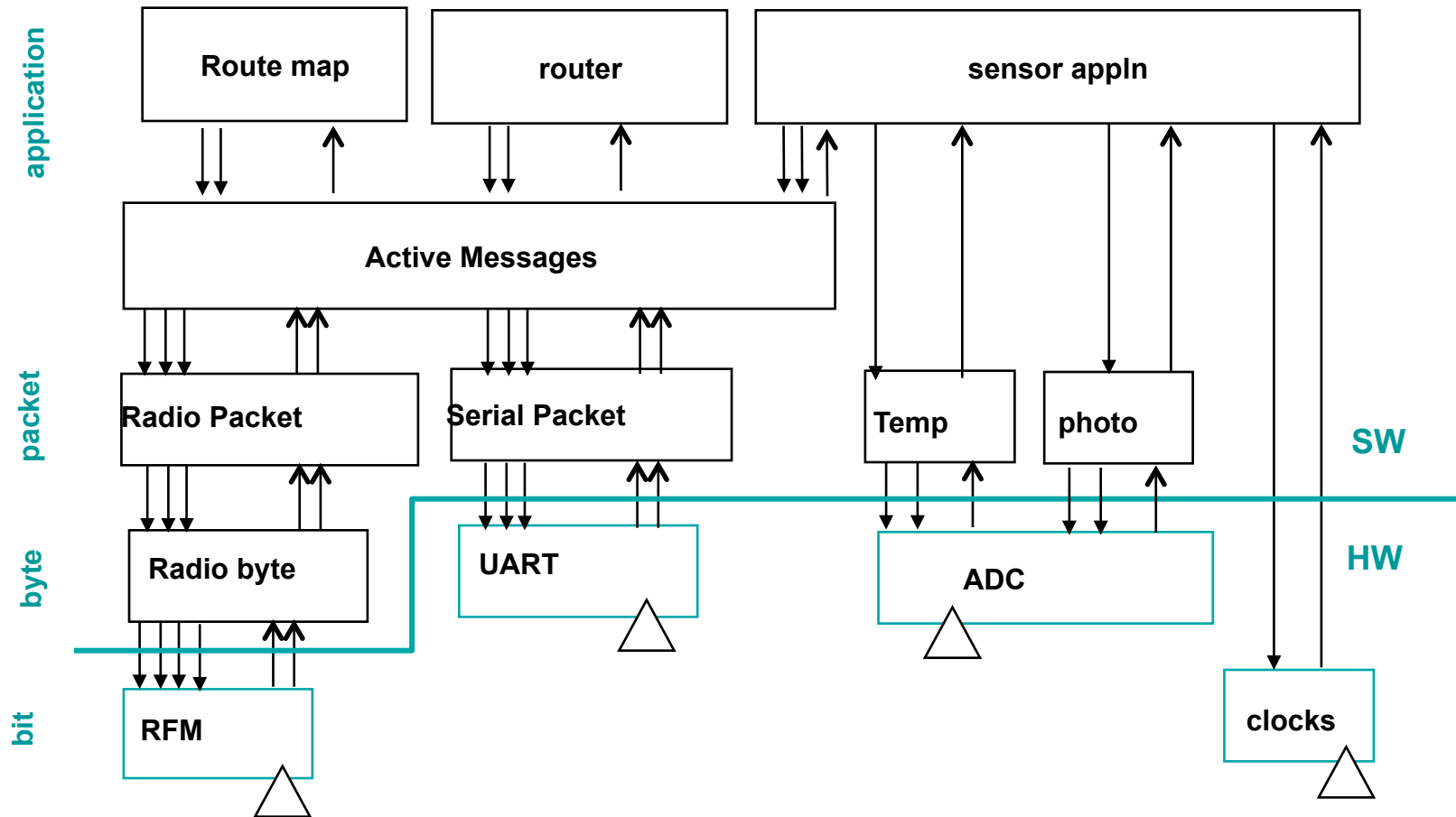


TOS Execution Model

- commands request action
 - *ack/nack at every boundary*
 - call cmd or post task
- events notify occurrence
 - HW intrpt at lowest level
 - may signal events
 - call cmds
 - post tasks
- Tasks provide logical concurrency
 - preempted by events



Application = Graph of Components



Programming Hardware

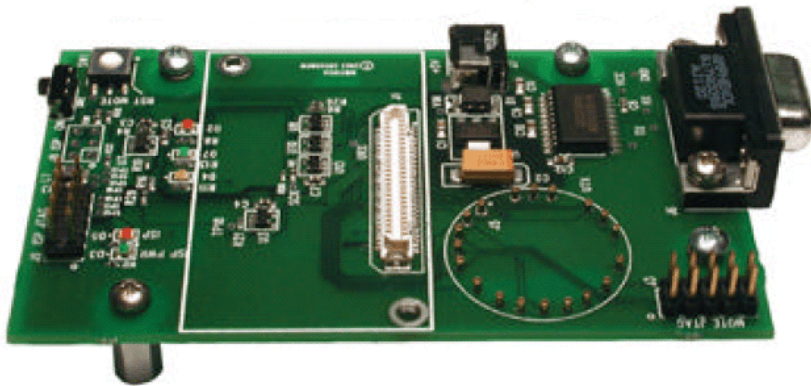
Mica2Dot



Mica Mote



Telos



MIB510 Serial Programming Board



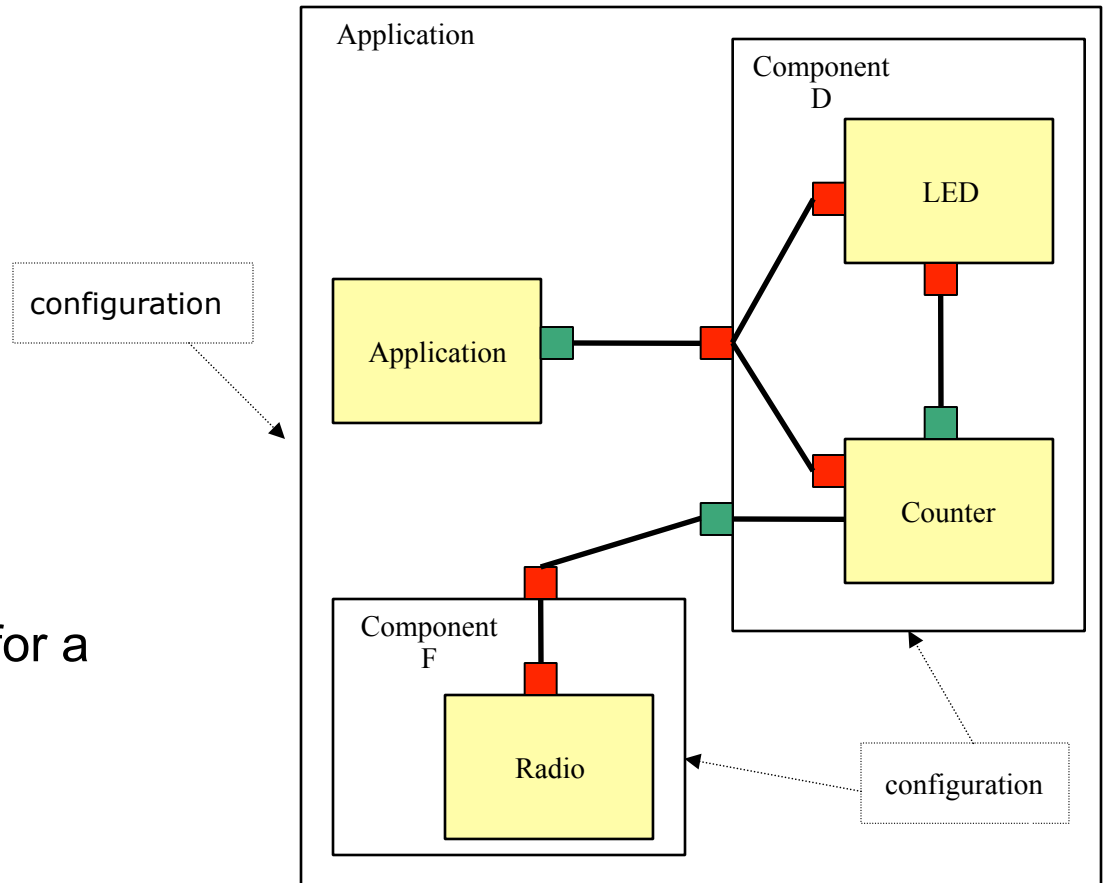
MIB600 Ethernet Programming Board

nesC: A programming language for sensor networks

- Designed to embody the structuring concepts and execution model of TinyOS
- Concurrency model
 - Non-blocking operations - *split phase*
 - Very low overhead, no threads
- Dialect of C with support for components
 - Components *provide* and *require* interfaces
 - Applications by wiring together components using configurations
- Whole program compilation and analysis
 - Inlining and race-detection
- Optimization approaches include
 - No function pointers
 - No dynamic memory allocation
 - No dynamic component instantiation/destruction

nesC

- the nesC model:
 - interfaces:
 - uses
 - provides
 - components:
 - modules
 - configurations
- application:= graph of components
- Why is this a good choice for a sensor net language?



Features of NesC

- small memory footprint +
- concurrency intensive application, event-driven architecture +
- power conservation +
- modular, easy to extend +
- good OS race conditions support. +
- simplistic FIFO scheduling -> no real-time guarantees -
- bounded number of pending tasks -
- no process management -> resource allocation problems -

Programming Environment

- download, install and build:
 - cygwin (<http://www.cygwin.com>)
 - WinAVR (<http://winavr.sourceforge.net>)
 - nesC (<http://nesc.sourceforge.net>)
 - Java JDK (<http://java.sun.com/j2se/1.4.1>)
 - **tinyOS distribution** (<http://sourceforge.net/projects/tinyos>)
- build your application
 - code your components
 - \$ make mica2 install.1
- debug your application with TOSSIM simulator:
 - \$ make pc
 - \$ build/pc/main.exe 25