CS 425 / ECE 428 Distributed Systems Fall 2015

Indranil Gupta

Sensor Networks

Lecture 24 A

Nov 12, 2015

Reading: Links on website

Some questions…

- What is the smallest transistor out there today?

Some questions…

- What is the smallest transistor out there today?
- A single Gold (Au) atom!
- Pentium P4 contains 42 M transistors
- Au atomic weight is 196 ~ 200.
- 1 g of Au contains 3 X 10^21 atoms => 7.5 X 10^18 Pentium P4 processors from a gram of Au => 1 billion P4's per person!

Some questions…

- How would you “monitor”:

a) a large battlefield (for enemy tanks)?
b) a large environmental area (e.g., movement of whales)?
c) your own backyard (for intruders)?

Sensors!

- Coal mines have always had CO/CO2 sensors
  - "Canary in a coal mine."
- Industry has used sensors for a long time, e.g., along assembly lines

Today…

- Excessive Information
  - Army needs to know about enemy troop deployments
  - Environmentalists collecting data on an island
  - Humans in society face information overload
- Sensor Networking technology can help filter and process this information (And then perhaps respond automatically?)

Harvard's deployment – Tungurahua volcano, Ecuador

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

Source: http://fiji.eecs.harvard.edu/Volcano

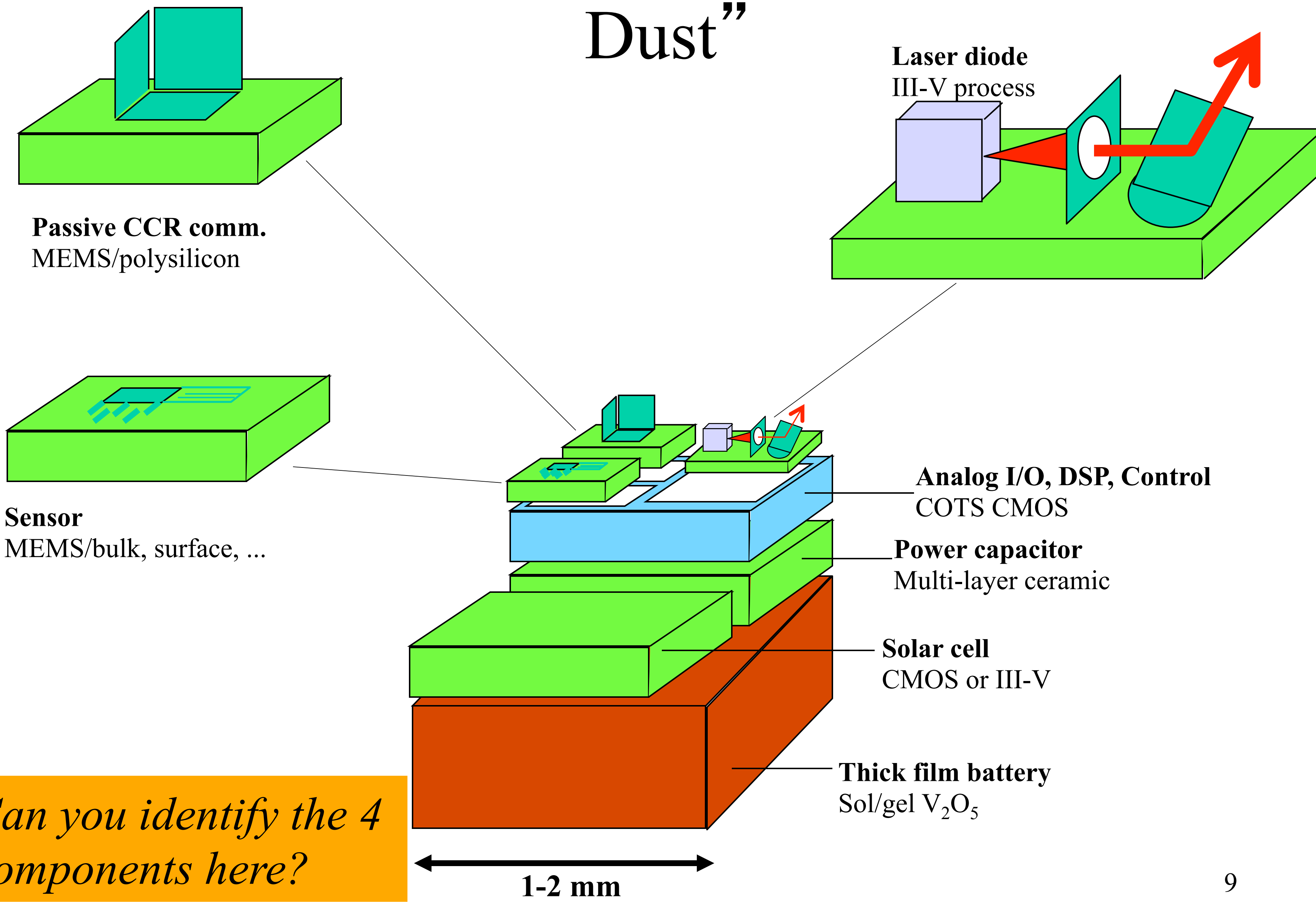
Growth of a technology requires

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 systems")
- 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"



Can you identify the 4 components here?

Example Hardware

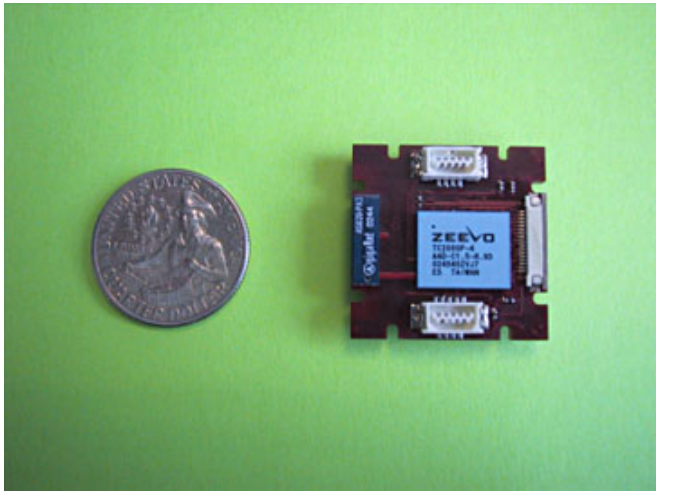
- Size
  - Golem Dust: 11.7 cu. mm
  - **MICA** motes: Few inches
- Everything on one chip: micro-everything
  - processor, transceiver, battery, sensors, memory, bus
  - MICA: 4 MHz, 40 Kbps, 4 KB SRAM / 512 KB Serial Flash, lasts 7 days at full blast on 2 x AA batteries

Examples

Spec, 2003

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

MICA: Crossbow



Similar Intel motes

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

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

- 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 => RF not good

  Instead, use *Optical transmission*: simpler hardware, lower power (think “laser”)
  - Directional antennas only, broadcast costly
  - Line of sight, or use intermediate node(s) to reflect (think “mirrors”)
  - Passive transmission (reflectors) => wormhole routing
  - However, switching links costly : mechanical antenna movements
  - Unidirectional links

Berkeley Family of Motes

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

Hardware Summary: Sensor Node

- Small Size : few mm to a few inches
- Limited processing and communication
  - MhZ clock, MB flash, KB RAM, at most 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
  - Meet real-time deadlines?
- Power consumption
- Communication

TinyOS design point

- – Bursty dataflow-driven computations
- – Multiple data streams => concurrency-intensive
- – Real-time computations
- – Power conservation
- – Size
- – Accommodate diverse set of applications (reuse mote)

TinyOS:

- – Event-driven execution (*reactive* mote)
- – Modular structure (components) and clean interfaces

Programming TinyOS

- Use a variant of C called NesC (Nested C)
- 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 (i.e., device driver for, e.g., the clock or the LED), or to a user-defined software module (e.g., routing)

    Modules implement and use *interfaces*
  - Or a *configuration* , a set of other components *wired* (virtually) together by specifying the unimplemented method invocations
- A complete NesC application then consists of one top level configuration

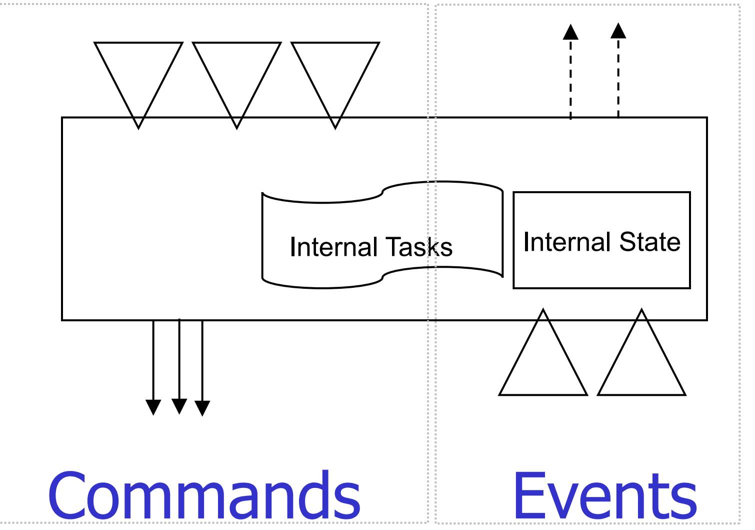
Steps in writing and installing your NesC app

(applies to MICA Mote)

- On your PC
  - Write NesC program
  - Compile to an executable for the mote
  - Debug on your PC (using TOSSIM)
  - Plug the mote into the PC through a connector board
  - Install the program
- On the mote
  - Turn the mote on, and it’s already running your application

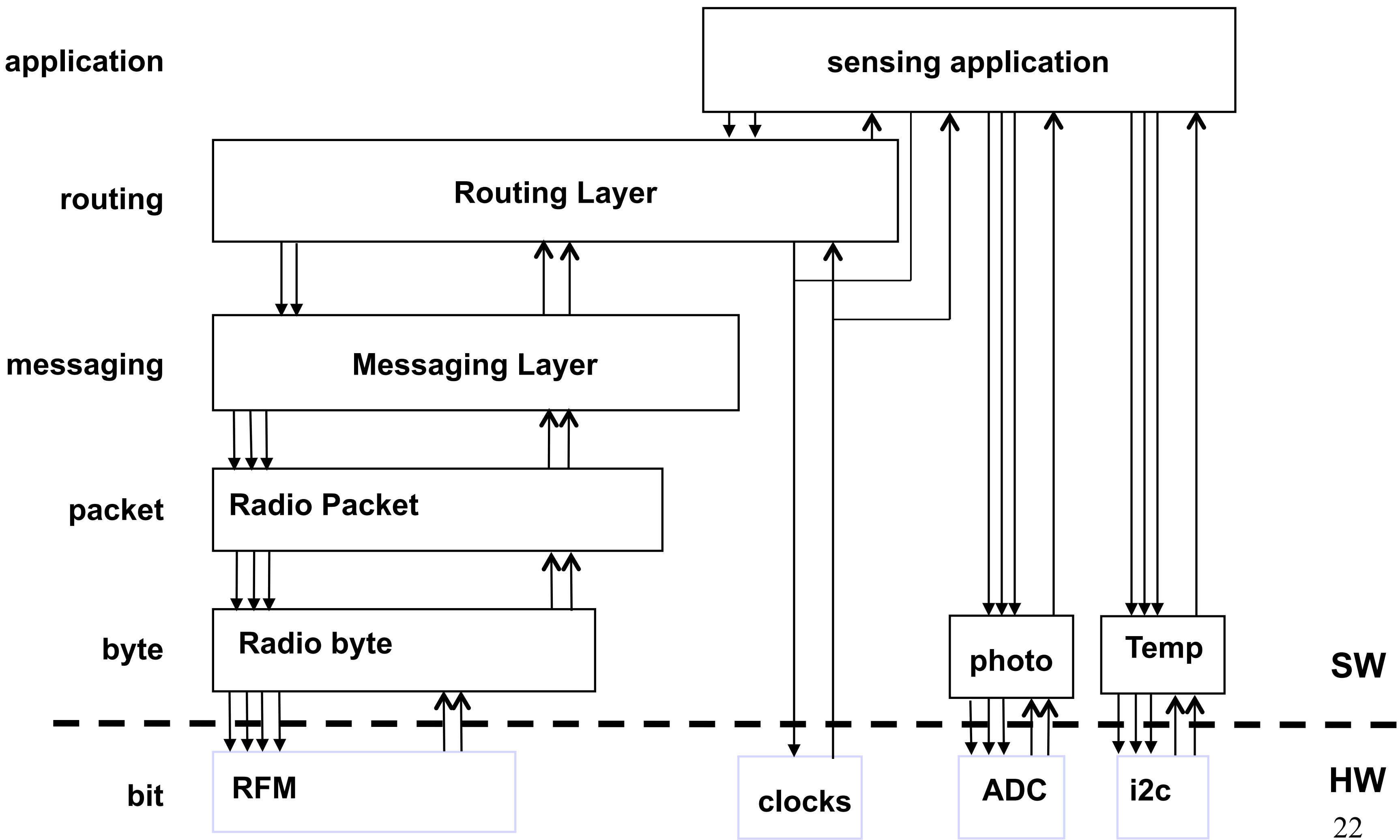
TinyOS component model

- Component specifies:



- Component invocation is event driven, arising from hardware events
- *Static allocation* of objects avoids run-time overhead
- Scheduling: dynamic, 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 µJ/bit
  Inactive State: 5 µA
  Peak Load: 20 mA
- Concurrency support: at peak load CPU is asleep 50% of time
- Events propagate through stack <40 µS

Energy – a critical resource

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

Energy – a critical resource

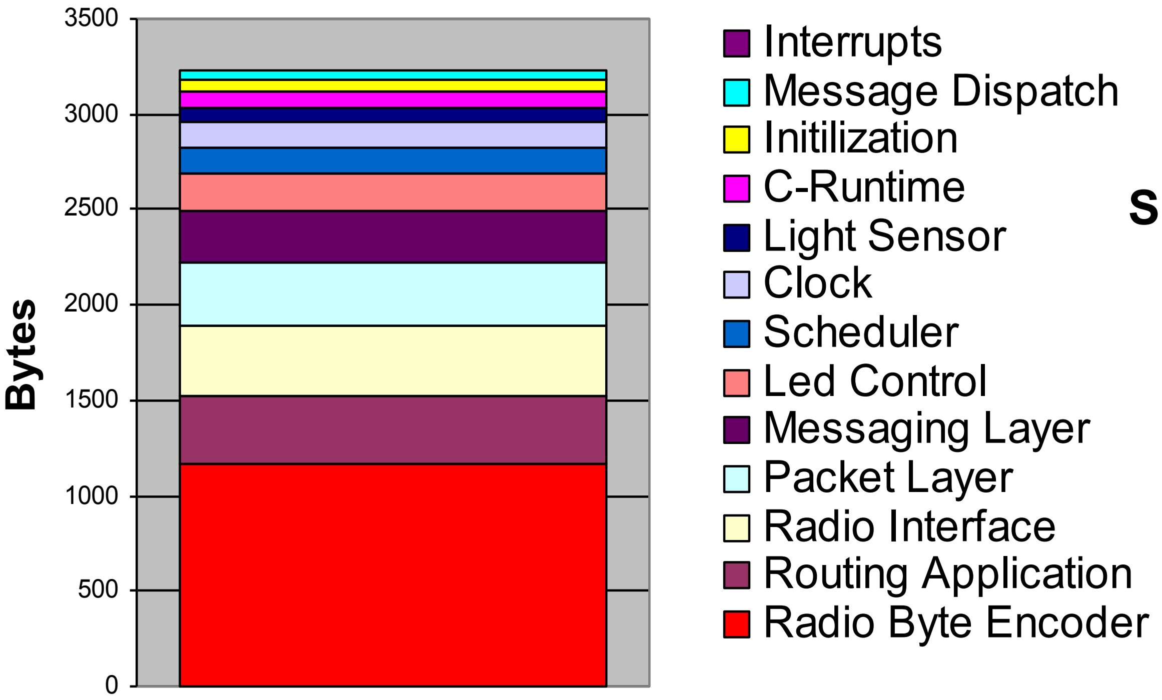
Component	*Rate*	*Startup time*	*Current consumption*
CPU Active	4 MHz	N/A	4.6 mA
CPU Idle	4 MHz	1 us	2.4 mA
CPU Suspend	32 kHz	4 ms	10 uA
Radio Transmit	40 kHz	30 ms	12 mA
Radio Receive	40 kHz	30 ms	3.6 mA
Photo	2000 Hz	10 ms	1.235 mA
I2C Temp	2 Hz	500 ms	0.150 mA
Pressure	10 Hz	500 ms	0.010 mA
Press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



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 (compared to say Linux)
  - TinyOS: efficient network interfaces and power conservation
  - Importance? Failure of a few sensor nodes can be made up by the distributed protocol
- @ Level of Protocols (for **data aggregation**)
  - Don't send raw data to base station
    - Too much power consumed
  - 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: nodes build a spanning tree using their neighbors and route data towards root; intermediate nodes aggregate data from children and send to parent
      - simple but will partition on failures
    - Better: denser graph (e.g., DAG) - more robust, but more expensive maintenance, and worry about double-counting values

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 if possible from s/w to h/w

@ Application-level ?

- Need: Applications written with scalability in mind

@ 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
- Event-driven model to the extreme: Asynchronous VLSI
  - No system clock on motherboard!
- Need: Self-sufficient (and self-stabilizing) sensor networks
  - In-network processing, monitoring, and healing
- Need: Scheduling
  - Of Computations across networked nodes
- Need: Security, and Privacy
- Need: Protocols for **anonymous** sensor nodes
  - No IP/MAC addresses, so need random addresses

Other Projects

- Berkeley
  - TOSSIM (+TinyViz)
    - TinyOS simulator (+ visualization GUI)
  - TinyDB
    - Querying a sensor net like a database table (every row ~ one sensor)
  - Maté, Trickle
    - Virtual machine for TinyOS motes, code update propagation in sensor networks for automatic reprogramming.
- 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
- Retinal prosthesis chips