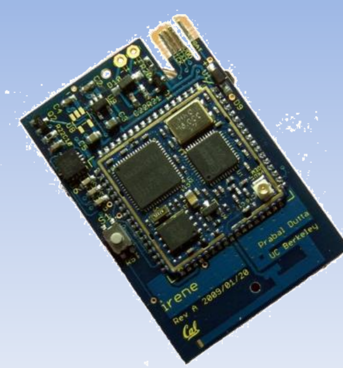
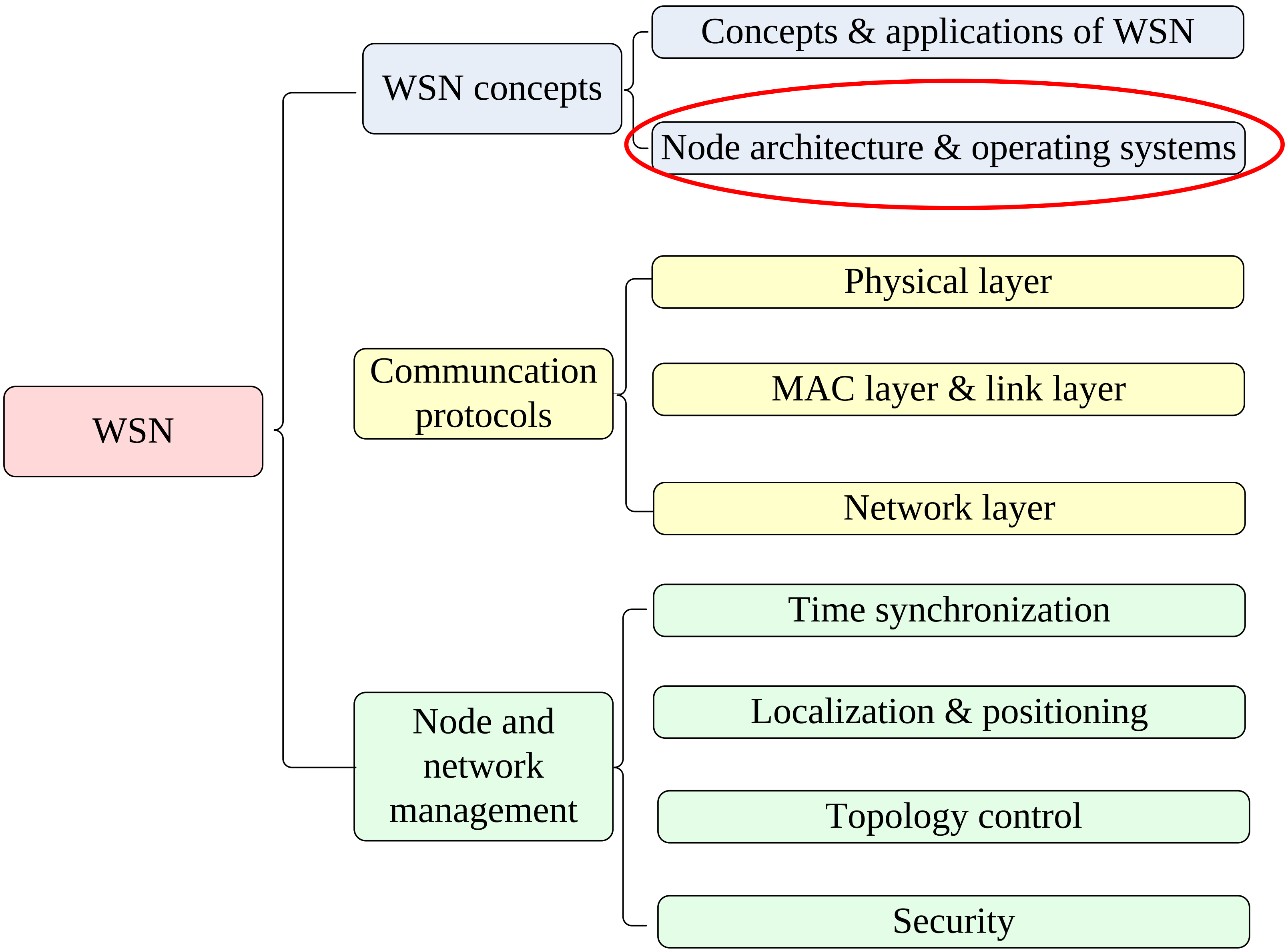
Wireless Sensor Networks

Lecture 2: Node architecture & operating system



Lecturer： Zhuo Sun
Office： 509 School of Computer Science
Email: zsun@nwpu.edu.cn

Course structure



Outline

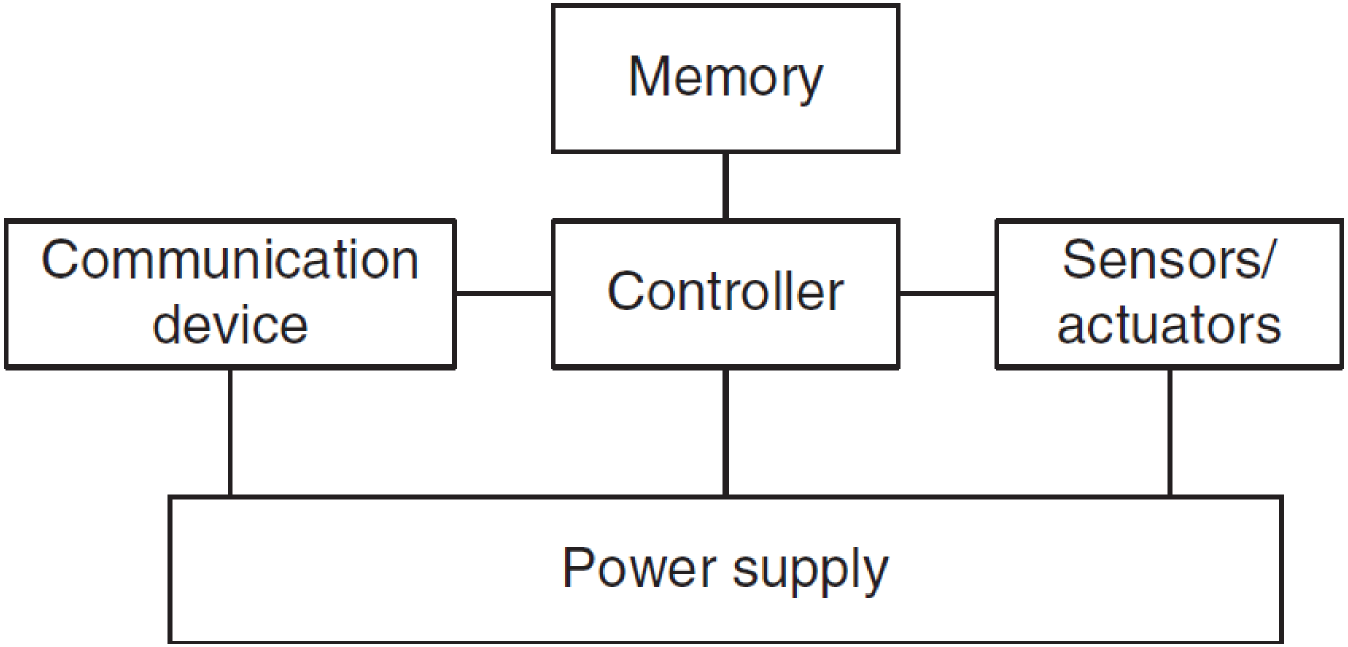
- Single node architecture
  - Hardware components
  - Energy supply and consumption
- Operating system
  - Runtime environment
  - Case study: TinyOS

Outline

- Single node architecture
  - Hardware components
  - Energy supply and consumption
- Operating system
  - Runtime environment
  - Case study: TinyOS

Hardware components

- Main components of a WSN node
  - **Sensors and actuators**: observe or control environment
  - **Controller**: process data and execute instructions
  - **Memory**: store programs and intermediate data
  - **Communication**: send and receive information
  - **Power supply**: provide energy and obtain energy



Controller

- Potential candidates
  - **Digital signal processors (DSPs)**: process large amounts of vectorial data
  - **Field-programmable gate arrays (FPGAs)**: reprogrammed for the changing set of requirements
  - **Application-specific integrated circuits (ASICs)**: less flexible but better energy efficiency/performance
  - **Microcontroller**: high flexibility in connecting devices and programming, built-in memory, low power consumption, time-critical signal processing

Controller

- Typical microcontrollers for WSN
  - **Texas Instruments MSP 430**
    - 16-bit Reduced Instruction Set Computer (RISC) core, up to 4MHz clock frequency, varying amount of on-chip Random Access Memory (RAM), several DAC, prices start at 0.49 US$
    - https://www.ti.com/microcontrollers-mcus-processors/microcontrollers/msp430-microcontrollers/overview.html

  - **Atmel Atmega**
    - 8-bit controller, larger memory than MSP430, extended instruction set, event system
    - https://www.microchip.com/en-us/product/ATMEGA328

西北工業大學 NORTHWESTERN POLYTECHNICAL UNIVERSITY

Communication device

- Communication medium
  - **Light:** line of sight between source and sink is required, short coverage range
  - **Ultrasound:** limited communication range
  - **Radio frequencies:** relatively long range, not require line of sight, acceptable error rates

- Radio transmitters send a bit stream as radio wave; Radio receivers receive and convert the radio wave into bit stream

Devices performing the two tasks are **transceiver**

Communication device

- Transceiver characteristics
  - Capabilities
    - Interface: packet, byte, bit level
    - Multiple channels: match application requirements
    - Data rates: tens of kilobits/second (low compared to cellular)
    - Range: at least tens of meters
  - Energy characteristics
    - Power consumption to send/receive data/ change states
    - Transmission power control, Power efficiency
  - Radio performance
    - Channel coding/Modulation techniques

Communication device

- Transceiver operational modes
  - Transmit
  - Receive
  - Idle
    - ready to receive but not doing so; some functions in hardware are switched off and reduce energy consumption
  - Sleep
    - significant parts of the transceiver are switched off; not able to immediately receive data
    - **Recovery time/startup energy** to leave sleep are significant

Question: Sleep mode is more energy efficient than non-sleep mode?

Communication device

- Wakeup receivers
  - Wake up without needing a significant amount of power
  - Simple solution: wakeup would happen for every packet
  - Improved solution: using proper address information to determine whether the incoming packet is actually for this node

Communication device

- Ultra-wideband communications
  - Standard narrow-band transceivers: modulate a signal onto a single carrier wave
  - Use a large bandwidth,
    - modulate a signal onto multiple carrier waves
    - do not modulate, simply emit a “burst” of power (almost rectangular pulse, very short, Information is encoded by using the presence/absence of pulses)

Communication device

- Ultra-wideband communications
  - Advantages
    - Pretty resilient to multi-path propagation
    - Very good ranging capabilities
    - Good wall penetration
  - Disadvantages
    - Requires tight time synchronization of receiver
    - Relatively short range (typically)

Sensors

- Main categories
  - Any energy radiated? Passive vs. active sensors
  - Directional sense? Directional vs. omnidirectional sensors
- Examples
  - Passive, omnidirectional: light, thermometer, microphones
  - Passive, narrow-beam(directional): camera
  - Active: sonar or radar
- Important parameter: coverage area

Question

Question1: A WSN node includes which components:

A. Sensors/actuators

B. Controller

C. Communication devices

D. Memory

E. Power supply

Outline

- Single node architecture
  - Hardware components
  - Energy supply and consumption
- Operating system
  - Runtime environment
  - Case study: TinyOS

Energy supply and consumption

- Store energy: Batteries(traditional way)
  - Primary batteries, non-rechargeable
  - Secondary batteries, if there is energy harvesting
- Requirements for Batteries
  - **High** capacity at small weight/volume and low price (metric: energy per volume, J/cm3)
  - **Varying** capacity under load
  - **Low** self-discharge
  - **Efficient** recharging
  - Relaxation **exploitation** (self-recharging of an empty battery)
  - Voltage **stability** (by DC-DC conversion)

Energy supply and consumption

- Energy scavenging
  - How to recharge the battery?
    - A laptop: easy, plug into wall socket
    - A sensor: harvest energy from environment
- Ambient energy sources
  - Light: solar cells
  - Temperature gradients
  - Vibrations (mechanical energy)
  - Pressure variation
  - Air/liquid flow(in wind mills or turbines)

Energy supply and consumption

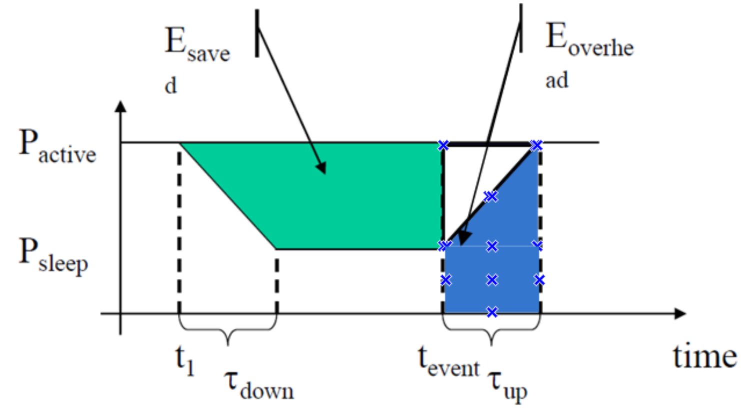
- Energy consumption for different operation states
  - If nothing to do, switch to power safe mode
  - Typical multiple modes
    - Controller: active, idle, sleep
    - Transceiver: turn on/off, transmitter, receiver
  - Not negligible time/energy to change modes, how to schedule the mode transitions?

Energy supply and consumption

- Controller energy consumption
  - Mode transition example (Event-triggered wake up from sleep mode

$$E_{\text{saved}} = (t_{\text{event}} - t_1) P_{\text{active}} - (\tau_{\text{down}} (P_{\text{active}} + P_{\text{sleep}})/2 + (t_{\text{event}} - t_1 - \tau_{\text{down}}) P_{\text{sleep}}).$$

$$E_{\text{overhead}} = \tau_{\text{up}} (P_{\text{active}} + P_{\text{sleep}})/2.$$



  - Switching to sleep mode is beneficial when

$$E_{\text{overhead}} < E_{\text{saved}}$$

$$(t_{\text{event}} - t_1) > \frac{1}{2} \left(\tau_{\text{down}} + \frac{P_{\text{active}} + P_{\text{sleep}}}{P_{\text{active}} - P_{\text{sleep}}} \tau_{\text{up}} \right)$$

Energy supply and consumption

- Controller energy consumption
  - Alternative: Dynamic voltage scaling
    - Scale voltage to adapt the computation speed to different tasks
    - Reduced voltage (lower clock rates, less speed), and less consumed power

$$P \propto f \cdot V_{\mathrm{DD}}^2$$

    - Benefits: discrete operational modes to continuous power adaptation, switching is easier
    - Cautions: operate based on the controller's specifications (maximum or minimum voltage level)

Energy supply and consumption

- Memory energy consumption
  - On-chip memory of a controller
    - Needed power is included in controllers' power consumption
  - FLASH memory
    - Reading and writing are expensive
    - Example: Flash memory on Mica node

      (reading: 1.1 nAh/byte; writing: 83.3 nAh/byte)

Energy supply and consumption

- Transmitter energy consumption
  - Amplifier power

$$P_{\text{amp}} = \alpha_{\text{amp}} + \beta_{\text{amp}} P_{\text{tx}}$$

  - Electronic components need power P_{txElec}
  - Transmitter energy consumption model

$$E_{\text{tx}}(n, R_{\text{code}}, P_{\text{amp}}) = T_{\text{start}} P_{\text{start}} + \frac{n}{R R_{\text{code}}} (P_{\text{txElec}} + P_{\text{amp}})$$

Startup energy

Time to transmit n-bits packet

***Tips*:** modulation and antenna efficiency are not considered.

Energy supply and consumption

- Receiver energy consumption
  - Transmitter energy consumption model

$$E_{\mathrm{rcvd}} = \underbrace{T_{\mathrm{start}} P_{\mathrm{start}}}_{\text{Startup energy}} + \underbrace{\frac{n}{R R_{\mathrm{code}}}}_{\text{Time to receive n-bits packet}} P_{\mathrm{rxElec}} + n \underbrace{E_{\mathrm{decBit}}}_{\text{Decoding power}}$$

Tips: Similar to DVS in the controller, it is promising to dynamically adapt modulation and coding for different channel gains, thereby maximizing energy efficiency and other system metrics(like throughput).

Energy supply and consumption

- Transceiver energy consumption example

Symbol	Description	Example transceiver		
		μAMPS-1 [559]	WINS [670]	MEDUSA-II [670]
α_{amp}	Equation (2.4)	174 mW	N/A	N/A
β_{amp}	Equation (2.4)	5.0	8.9	7.43
P_{amp}	Amplifier pwr.	179–674 mW	N/A	N/A
P_{rxElec}	Reception pwr.	279 mW	368.3 mW	12.48 mW
P_{rxIdle}	Receive idle	N/A	344.2 mW	12.34 mW
P_{start}	Startup pwr.	58.7 mW	N/A	N/A
P_{txElec}	Transmit pwr.	151 mW	≈ 386 mW	11.61 mW
R	Transmission rate	1 Mbps	100 kbps	OOK 30 kbps ASK 115.2 kbps
T_{start}	Startup time	466 µs	N/A	N/A

  - Considerable startup time/energy(transceiver and system architecture)
  - Comparable transmitting and receiving power

Energy supply and consumption

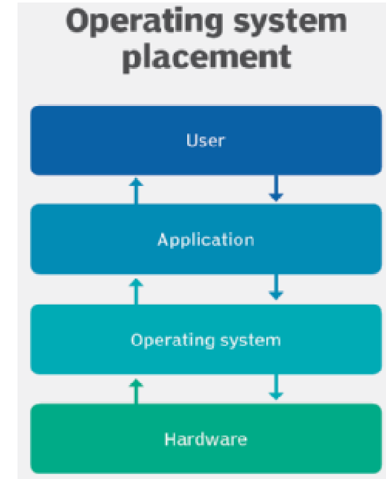
- Computation vs. communication energy cost
  - Comparison
    - Energy ratio of "sending one bit" to "computing one instruction" is between 220 and 2900 for different hardware
    - Communicating one kilobyte=computing three million instructions!
  - Solution
    - Try to compute instead of communication whenever possible
    - **In-network processing**

Outline

- Single node architecture
  - Hardware components
  - Energy supply and consumption
- Operating system
  - Runtime environment
  - Case study: TinyOS

Runtime environment

- Operating system challenges in WSN
  - General-purpose operating system goals
    - Make access to device resources
    - Manage resources from concurrent access
  - General-purpose operating system means
    - Protected operation modes of the CPU
    - Support by a memory management unit
  - Problems for microcontrollers
    - More restricted executing codes: partial tasks are required
    - Low cost and energy efficiency: no separate protection modes and memory management unit

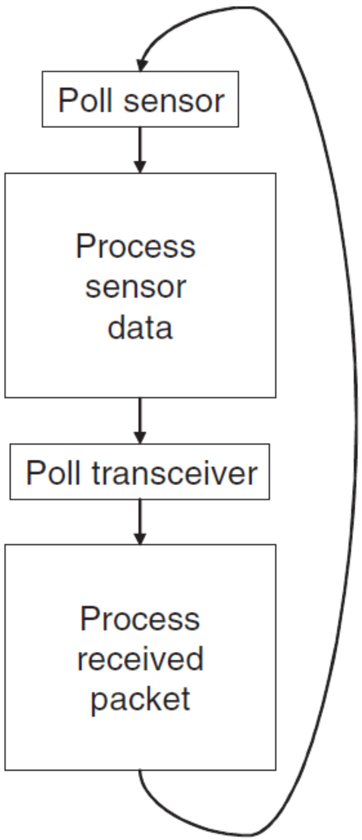


Runtime environment

- Potential operating system in WSN
  - Characteristics of WSN nodes
    - Only a single “application” running on a WSN node
    - No need to protect malicious software parts from each other
    - Direct hardware control by application is efficient
  - No OS, only a simple runtime environment
    - Appropriate programming model
    - Clear protocol stack structure
    - Explicit energy management support

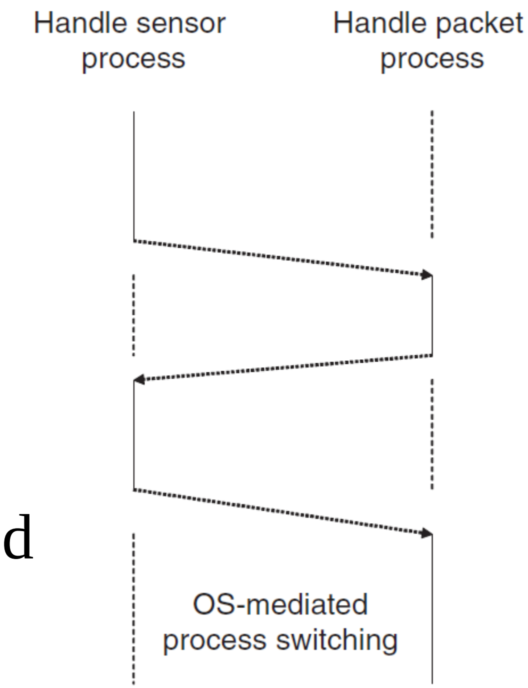
Runtime environment

- Appropriate programming model
  - Concurrency is needed
    - A WSN node has to handle multiple sensors' data, perform computation for application, and execute communication software
  - Sequential programming model
    - Poll one by one
    - Risk of missing sensors' data when processing a packet



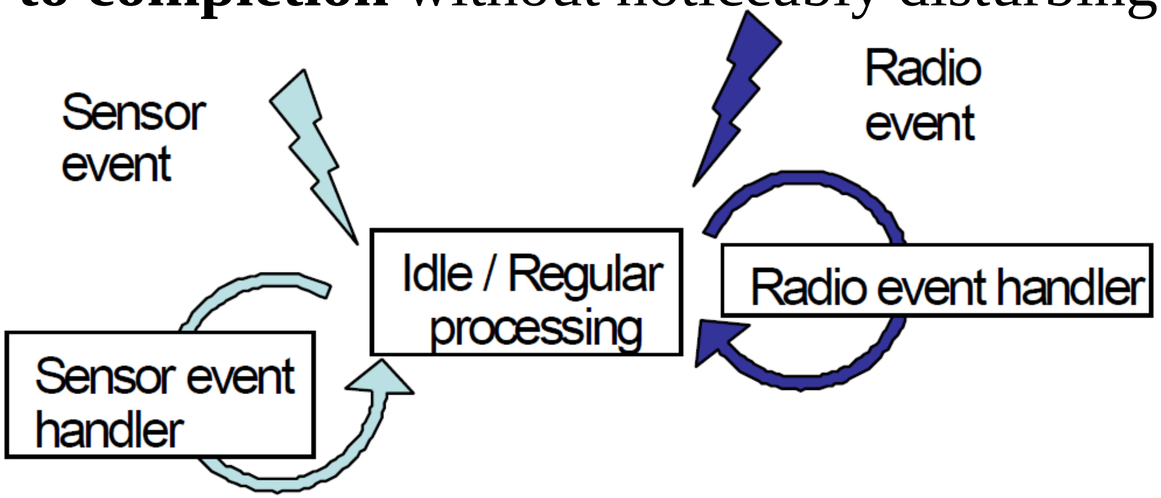
Runtime environment

- Appropriate programming model
  - Process-based concurrency (general-purpose)
    - Parallel execution of multiple processes
    - Based on interrupts and context switching
    - But: several tasks are executed and are small with respect to switching overhead; protection between processes is not needed



Runtime environment

- Appropriate programming model
  - Event-based programming
    - Two contexts: one for **event handlers**, one for **regular execution**
    - store data and post information that the event has happened
    - Perform regular processing or be idle
    - React to events when it is interrupted by event handlers
    - **Run to completion** without noticeably disturbing other codes



Runtime environment

- Protocol stack structure
  - Layered approach (OSI model)
    - Benefits: manageable, containing complexity, promoting modularity and reuse
    - But: less flexible, no cross-layer information exchange
  - Component approach
    - Component: a single and well-defined function
    - Access to each other
    - Fit with event-based programming model

Runtime environment

- Dynamic energy management
  - When to switch to power-safe mode
    - Greedy sleeping is not beneficial (time/energy overhead)
    - Example: model the probability of next event happening and select most power-safe mode
  - How to control dynamic voltage scaling
    - Task deadlines bound the required speed
    - Several tasks with various deadlines in an operating system
  - Tradeoff fidelity against energy consumption
    - More energy consumed, more accurate results obtained

Case study

- TinyOS and nesC
  - TinyOS developed by UC Berkely as a runtime environment for their “motes”
  - nesC as programming language
  - Most important design aspects:
    - Component-based system
    - Components interact by exchanging asynchronous events
    - Components form a program by wiring them together(akin to VHDL, a hardware description language)

Conclusions

1. Five components of a node hardware architecture
2. Energy supply and consumption model of a WSN node
3. Event-based and component-based runtime environment
4. One example of this runtime environment-TinyOS

Assignment

Assignment: Please summarize the methods to improve the energy efficiency, from the view of node architecture design.