Power Considerations for Sensor Networks

Mani Srivastava
UCLA

In collaboration with:

USC/ISI (Bob Parker, Brian Schott)
Rockwell Science Center (Charles Chien)

Outline

- Power analysis of some sensor nodes
- Energy-efficient multihop packet forwarding
- Power-aware real-time processing

Power analysis of sensor nodes: Where does the power go?

- **High-end sensor node: Rockwell WINS nodes**
  - **StrongARM processor**
  - **Connexant's RDSSS9M 900MHz DECT radio (128 kbps, ~ 100m)**
  - **Seismic sensor**
- **Low-end sensor node: Experimental node similar to Berkeley's COTS motes**
  - **Atmel AS90LS8535 microcontroller**
  - **RF Monolithic's DR3000 radio (2.4, 19.2, 115 kbps, ~ 10-30m)**
  - **No sensors (but microcontroller has ADC)**

Power Analysis of Rockwell's WINS Nodes (Measurements)

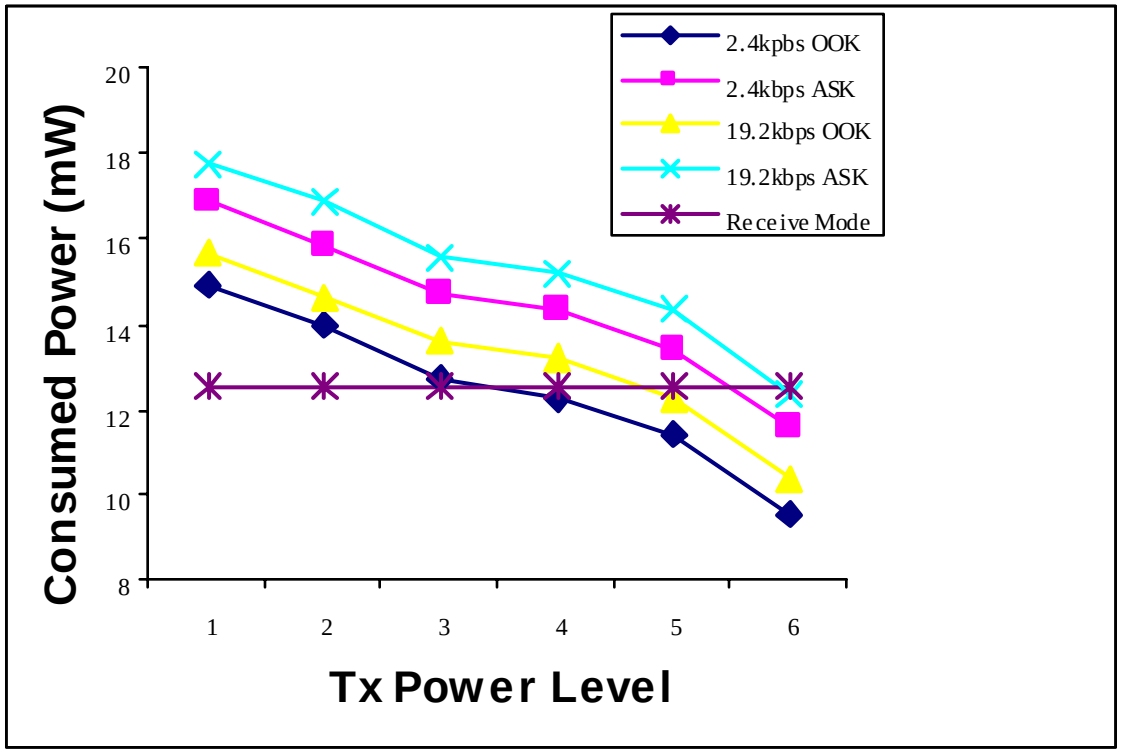
Processor	Seismic Sensor	Radio	Power (mW)
Active	On	Rx	751.6
Active	On	Idle	727.5
Active	On	Sleep	416.3
Active	On	Removed	383.3
Sleep	On	Removed	64.0
Active	Removed	Removed	360.0
Active	On	Tx (36.3 mW)	1080.5
		Tx (27.5 mW)	1033.3
		Tx (19.1 mW)	986.0
		Tx (13.8 mW)	942.6
		Tx (10.0 mW)	910.9
		Tx (3.47 mW)	815.5
		Tx (2.51 mW)	807.5
		Tx (1.78 mW)	799.5
		Tx (1.32 mW)	791.5
		Tx (0.955 mW)	787.5
		Tx (0.437 mW)	775.5
		Tx (0.302 mW)	773.9
		Tx (0.229 mW)	772.7
		Tx (0.158 mW)	771.5
		Tx (0.117 mW)	771.1

Summary

- **Processor**
  - **Active = 360 mW**
    - **doing repeated transmit/receive**
  - **Sleep = 41 mW**
  - **Off = 0.9 mW**
- **Sensor = 23 mW**
- **Processor : Tx = 1 : 2**
- **Processor : Rx = 1 : 1**
- **Total Tx : Rx = 4 : 3 at maximum range**
  - **comparable at lower Tx**

Power Analysis of Experimental Node (Measurements)

Mode	Power Level	OOK @ 2.4 kbps	OOK @ 19.2kbps	ASK @ 2.4 kbps	ASK @ 19.2kbps
Tx	0.7368	14.88	15.67	16.85	17.76
Tx	0.5506	13.96	14.62	15.80	16.85
Tx	0.3972	12.76	13.56	14.75	15.54
Tx	0.3307	12.23	13.16	14.35	15.15
Tx	0.2396	11.43	12.23	13.43	14.35
Tx	0.0979	9.54	10.35	11.56	12.36
Rx		12.5	12.50	12.50	12.50
Idle		12.36	12.36	12.36	12.36
Sleep		0.016	0.016	0.016	0.016



Note

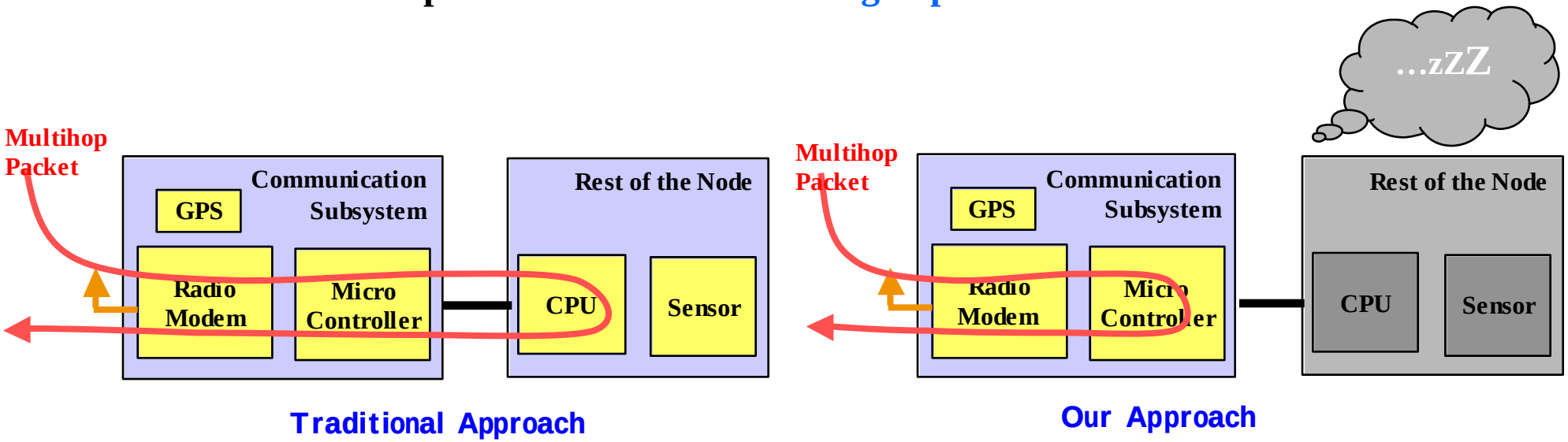
- All powers in mW
- Microcontroller (with ADC)
  - Active = 8.7 mW
  - Idle = 5.9 mW
  - Off = 3 μW

Some Observations from Power Analysis

- In WINS node, radio consumes 33 mW in “sleep” vs. “removed”
  - Argues for module level power shutdown
- Tx and Rx power
  - Rx power within 40% of maximum Tx power
  - Under certain circumstances, Tx power < Rx power!
  - Argues for:
    - MAC protocols that do not “listen” a lot
    - Low-power paging (wakeup) channel
- Processor power fairly significant (30-50%) share of overall power
- Sensor transducer power negligible
  - Use sensors to provide wakeup signal for processor and radio

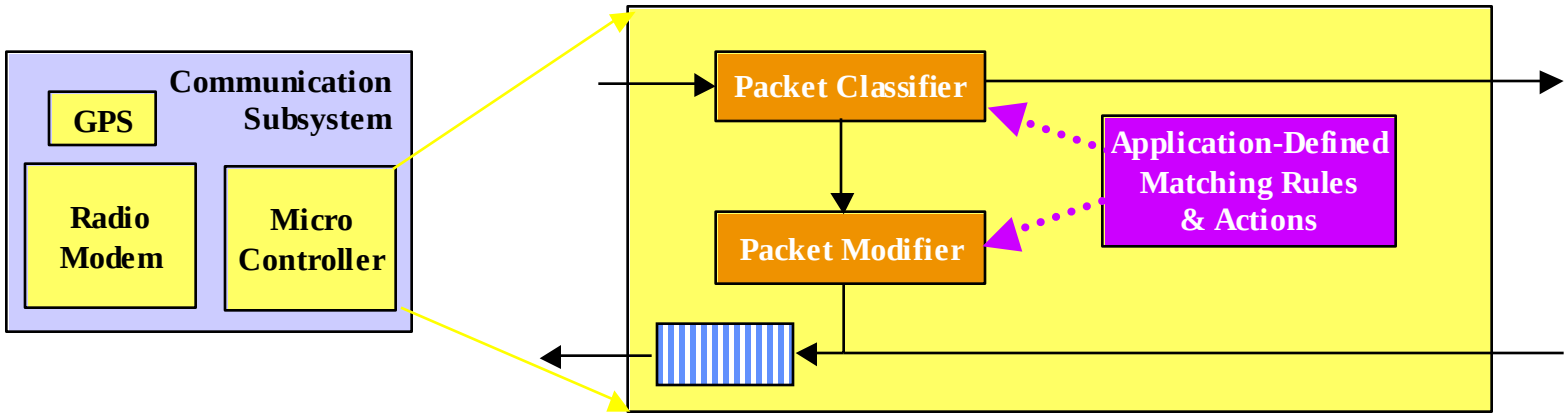
Energy-efficient Multihop Packet Forwarding Architecture

- **Problem: radios often simply relay packets in multihop network**
- **Traditional approach: main CPU woken up, packets sent to it across serial bus**
  - **power hungry computing and communication operations**
- **Our approach: exploit programmable micro-controller in the Communication Subsystem to handle common cases of packet routing**
  - **can also do operations such as combining of packets with redundant information**



- **Key challenge: how to do it so that every new routing protocol will not require a new radio firmware**
- **Solution: *application-defined all-layer packet forwarding***

Application-defined All-layer Packet Forwarding



- Packet-classifier and packet-modifier driven by application defined matching rules and actions
  - Matching rules: and/or expressions using =, <, >, range operators on arbitrary packet fields (offset, length)
  - Actions: accept, forward, drop, field increment/decrement etc.
- Rules and actions operate on arbitrary packet fields (any layer)
  - fields specified as (offset, length)
  - only simple, common cases handled at the radio
    - for complex cases packet sent to the main processor
- Expressiveness: implemented the following as test cases
  - Node ID-based addressing and routing (IP-like)
  - Point-cast (send to a rectangular geographical area specified as destination)
- Current proof-of-concept prototypes
  - Rockwell node
  - Experimental platform using Triscend's microcontroller with on-chip FPGA

Power-aware Real-time Processing with Energy-fidelity Trade-off

- Question: what kind of dynamic power management techniques makes sense on the CPU of the sensor node?
  - Software typically organized as RTOS with priority-based preemptive scheduling
    - Typically static priority
    - eCos, uCos, Neutrino, WinCE etc.
- Traditional approaches of power management aren't the best:
  - Select a (fixed) lower voltage when designing the board
    - Can't handle tight deadlines
  - Shutdown whenever idle
    - Only gets a linear improvement
  - Example: consider a task set {(10, 3, 10), (14, 7, 14)}
    - CPU utilization is 80%
    - Shutdown will save 20% power
    - Can't slow CPU by 20% (& reduce V) as deadlines no longer met
    - Can do better by combining static voltage scaling and shutdown (22.5% saving in this example)

Problem:
Traditional approaches use WCET (worst case execution time) and aim for no deadline misses

Reality #1: Significant Variation in Execution Times

- **WCET : BCET is typically >> 1, e.g.:**

Program	Description	BCET	WCET	WCET/BCET
DES	Data Encryption	73,912	672,298	**9.1**
DJPEG	JPEG decompression 128x96 color	12,703,432	122,838,368	**9.7**
FDCT	JPEG forward DCT	5,587	16,693	**3**
FFT	1024-point FFT	1,589,026	3,974,624	**2.5**
Matcnt	Summation of 2 100x100 matrices	1,722,105	8,172,149	**4.7**
Piksrt	Insertion sort of 10 elements	236	5,862	**24.8**
Sort	Bubble sort of 500 elements	13,965	50,244,928	**3598**
Stats	Sum, mean, var of 2 1000-size arrays	1,007,815	2,951,746	**2.9**

- **But, execution time variations in sensor data are not random**
  - **temporal correlation in underlying physical signal**
  - **can attempt to predict!**

Reality #2: Sensor Applications Tolerant to Deadline Misses

- Computation deadline misses lead to data loss
- Packet loss common in wireless links
  - e.g. a wireless link of 1E-4 BER means packet loss rate of 4% for small 50 byte packets
  - radio links in sensor networks often worse
- Significant probability of error in sensor signals
  - noisy sensor channels
- Applications designed to tolerate noisy/bad data by exploiting spatio-temporal redundancy
  - high transient losses acceptable if localized in time or space

If the communication is noisy, and applications are loss tolerant, is it worthwhile to strive for perfect noise-free computing?

Exploiting Execution-time Variation and Tolerance to Deadlines

- **Our strategy: predict execution time of task instance and dynamically scale voltage even more aggressively so as to minimize shutdown**
- **Execution time prediction**
  - **learn distribution of execution times (pdf)**
  - **Tasks with distinct modes can help the OS by providing hint after starting**
    - **E.g. MPEG decode can tell the OS after learning whether the frame is P, I, or F**
- **But, some deadlines are missed!**
- **Adaptive control loop to keep deadlines missed under control**
- **Typical result: 1.5-3x higher power saving compared to best conventional schemes with dynamic voltage, with < 1% deadlines missed**
- **Provides adaptive power-fidelity trade-off**

Power-aware RTOS Scheduler Implementation

- **RTOS predicts the remaining runtime (at max CPU speed) of a task instance**
  - **calculated whenever the task instance enters the system, or is preempted**
  - **based on run-times of previous instances of the task, and the run-time consumed so far**
    - **e.g. weighted mean**
    - **e.g. a coarse-grained discrete probability distribution of actual run time of each task is calculated, and used to calculate *E[remaining_runtime | runtime_so_far]***
  - **adaptively adjusts a multiplicative factor dependent on recent deadline misses**
- **Voltage scheduling strategy**
  - **Statically calculate a maximal global slowdown factor for the task set**
  - **Dynamically calculate a task-specific slowdown factor on context switch to stretch the task to its remaining WCET**
- **Results**
  - **Power savings of up to x2 (over best known DPM schemes) with deadline misses from < 1% to ~ 10% depending on task sets**

Current Status

- **Simulation tool for RTOS power management evaluation**
  - **PARSEC discrete event simulation language**
  - **Two communicating entities:**
    - **Task Generator**
      - **generates task instances with run times according to a trace or a distribution**
    - **RTOS**
      - **sets CPU speed by setting voltage and frequency**
      - **implements runtime predictor**
  - **Variety of task sets from literature**
  - **Note: non-predictive scheme is obtained by setting predictor to always return WCET – run time so far.**
- **Implementation in eCoS RTOS**
  - **Running on Intel StrongARM evaluation boardwith frequency variation (but no voltage variation)**