

System Architecture Directions for Networked Sensors[1]

Secure Sensor Networks Seminar presentation

Eric Anderson

Outline

- Sensor Network Characteristics
- Architectural Requirements
- Example hardware
- TinyOS
- Performance
- Architectural Implications

Sensor Network Characteristics

- *Power constrained*
- \Rightarrow Minimal hardware
 - Low-speed processor
 - Small RAM
 - Little or no controller hierarchy
- Real-time data flow(s)
- Application diversity
 - Customized, heterogeneous motes
- Physical maintenance impractical

System Requirements (1)

- Power-efficient operation
- Real-time operation
 - Limited storage \Rightarrow little buffering: Data must be processed as it arrives.
 - Limited controller hierarchy \Rightarrow CPU interfaces directly with devices.
- Concurrency-intensive
 - Multiple devices and data sources
 - All with real-time requirements
 - \Rightarrow Fast context switches required.

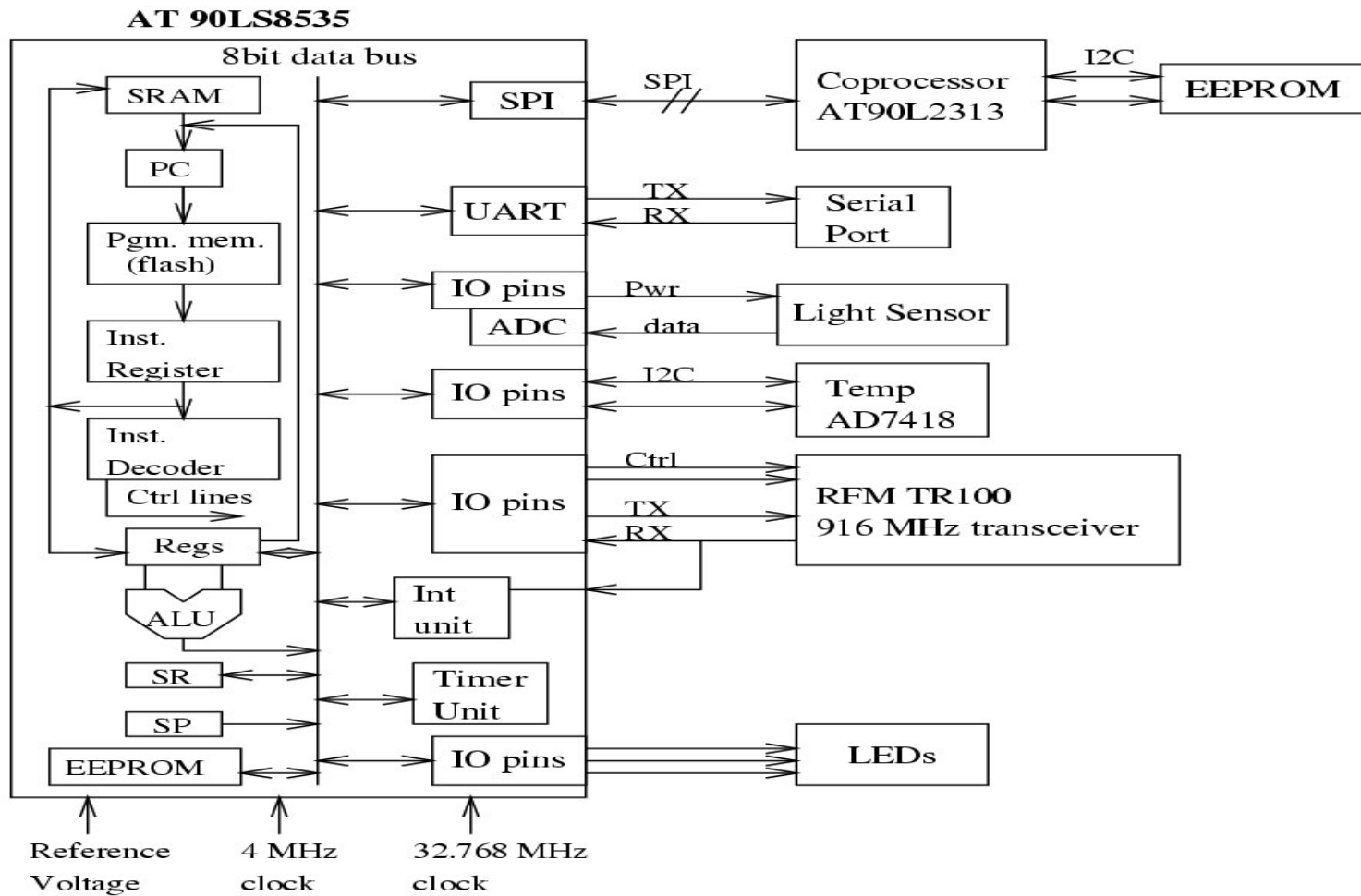
System Requirements (2)

- Very low (code & data) memory consumption
- Modularity
 - Motes are application-specific.
 - Will not require all “standard” OS components.
 - Will require custom components.
- Permeable HW/SW boundary
 - Allows application designers to make appropriate trade-offs.
 - System should offer similar abstraction for HW and SW modules.

Hardware Example

- CPU: 4MHz, 32 8-bit general-purpose registers
- Memory (code): 8KB flash
- Memory (data): 512B SRAM
- Devices directly controlled by MCU:
 - Temp & Light Sensors
 - Radio (on-off control, ≤ 19.2 Kbps)
 - LEDs
- Devices with dedicated controllers:
 - EEPROM (via SPI controller & coprocessor)
 - Serial port (via UART)

Schematic



Power Consumption - Revisited

- Battery provides $\approx 575\text{mAh}$ of energy.
- Power consumption ranges from 19.5mA at full power to $10\mu\text{A}$ in *inactive* mode. (Differ by factor of ≈ 2000 .)
- To send or receive data, system must be *active* or *idle* for radio min. pulse width ($52\mu\text{s}$).

Component	Active (mA)	Idle (mA)	Inactive (μA)
MCU core (AT90S8535)	5	2	1
MCU pins	1.5	-	-
LED	4.6 each	-	-
Photocell	.3	-	-
Radio (RFM TR1000)	12 tx	-	5
Radio (RFM TR1000)	4.5 rx	-	5
Temp (AD7416)	1	0.6	1.5
Co-proc (AT90LS2343)	2.4	.5	1
EEPROM (24LC256)	3	-	1

Concurrency - Revisited

- Radio drives concurrency requirements:
 - Want to communicate as quickly as possible to minimize wake time. (\Rightarrow use min. pulse width)
 - Radio requires service for each bit.
 - OS must service radio every $50\mu s$
 - If doing anything else, must context switch twice per $50\mu s$ (40,000 switches /s.)
 - At 4MHz, 100 cycle “budget” for switching overhead and work.
 - Copying 1 B (where to where?) requires 8 cycles.

TinyOS

OS Design Dimensions

- “Process” scheduling
- Component model
- ...

Processes and Scheduling (1)

TinyOS uses 2-level scheduler:

- *Events*
 - Events are triggered (directly or indirectly) by hardware.
 - Events run immediately.
 - Events may signal higher-level events and call *commands*.
 - Not interruptible?

Processes and Scheduling (2)

- *Tasks*
 - Tasks are longer-running computations.
 - Tasks are interruptible by events, but atomic w/r/t other tasks.
 - Tasks are *posted* by commands, and run to completion in FIFO order.
- Scheduler puts system into *idle* when task queue is empty.
- (Peripherals remain powered on, so hardware can trigger events)
- Application may manage power more aggressively.

Processes and Scheduling (3)

Std. Multi-threading	TinyOS Micro-threading
Atomic interrupt handlers	Atomic event handler
Interrupt preempts threads	Event preempts tasks
Interleaved thread execution	Task run-to-completion
1 stack per thread	1 stack (total) & 1 frame per component

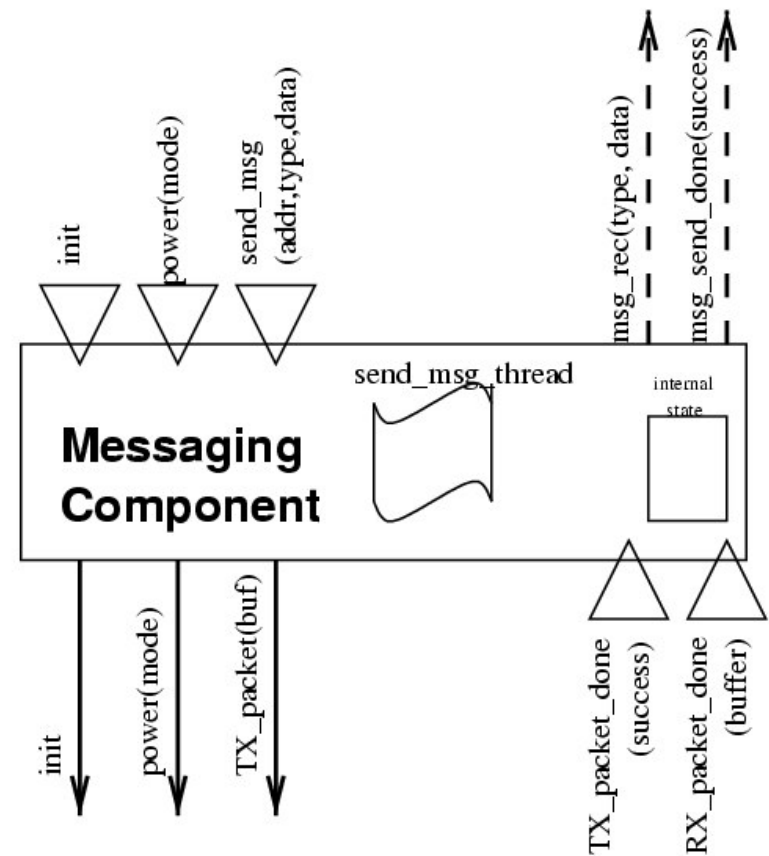
Components

Components are the basic software units in TinyOS.

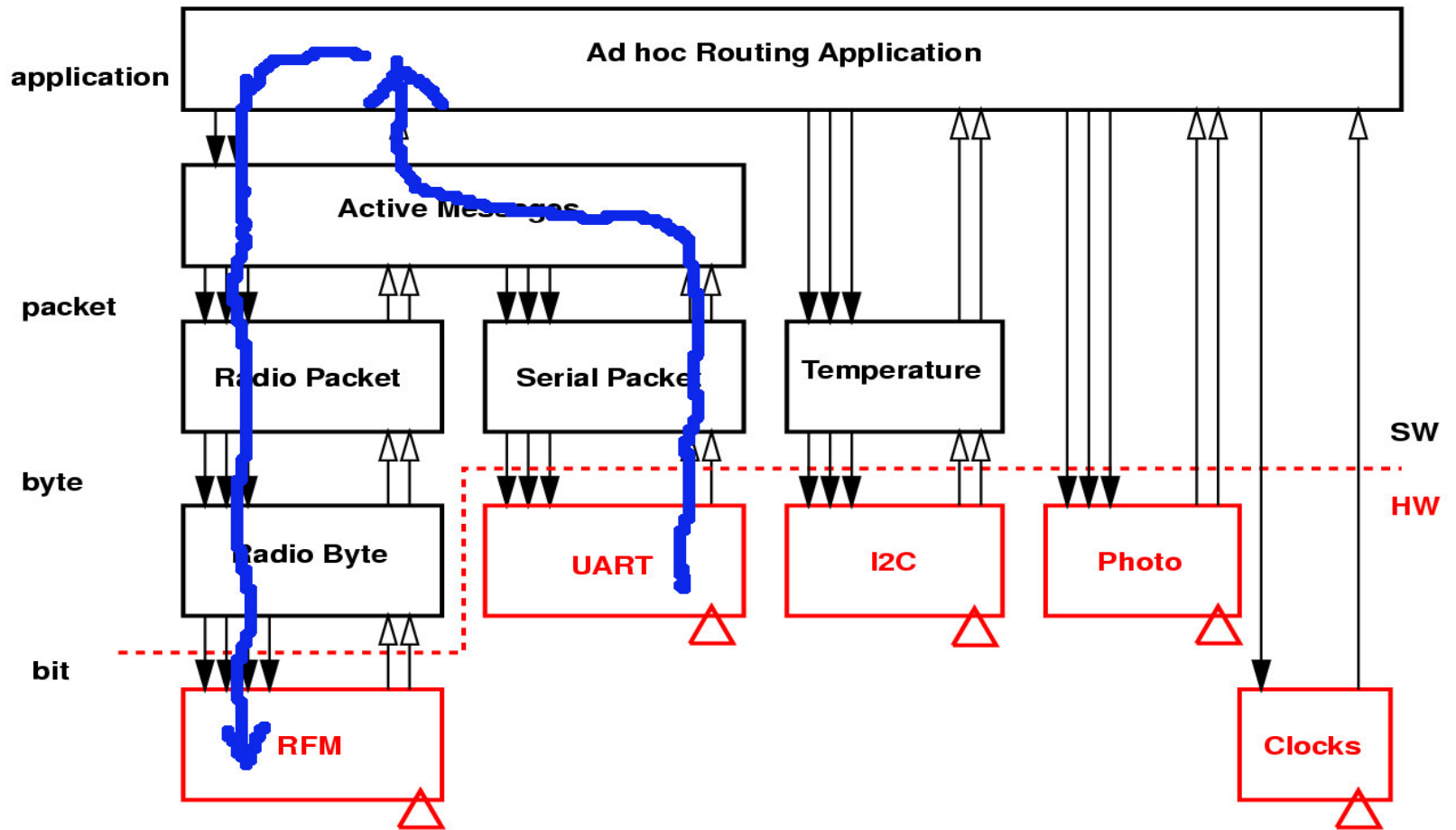
- Data
 - Statically-allocated fixed-sized *frame*.
- Code
 - Event handlers (act on frame)
 - Command handlers (act on frame)
 - Tasks (use stack (& acts on frame?))
- Declarations
 - Events signalled
 - Commands used

Component Interconnection

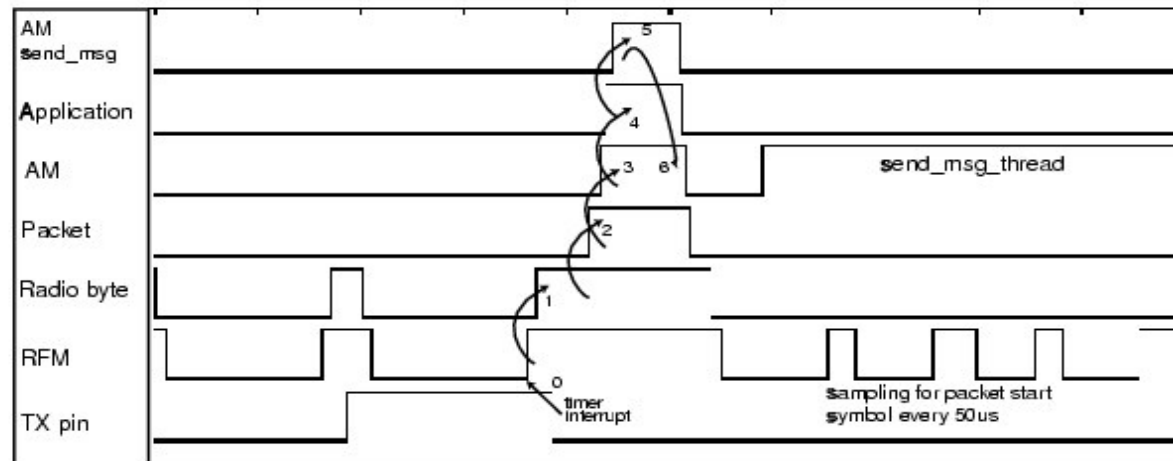
- Commands used by a component are provided by a “lower” one.
- Events signalled by a component are handled by a “higher” one.



Components & Execution Example



Components & Execution Timing



NOT the same execution as prev. slide

- 0-2 are events.
- 3 begins command, 4 ends.
- 5 is another command.
- 6 posts new task.

Performance

Operations	Cost (cycles)	Time (μ s)	Normalized to byte copy
Byte copy	8	2	1
Post an Event	10	2.5	1.25
Call a Command	10	2.5	1.25
Post a task to scheduler	46	11.5	6
Context switch overhead	51	12.75	6
Interrupt (hardware cost)	9	2.25	1
Interrupt (software cost)	71	17.75	9

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
Procesor_init	172	30
TinyOS scheduler	178	16
C runtime	82	0
Total	3450	226

Architectural Implications

- Single controller can handle multiple devices.
- (but) Bit-level radio is limited to about 40Kbps
- (51 cycles * 80,000 context switches / second = 4.08 MHz)
- Modular design allows “Radio Byte” SW component to be replaced with HW.
- Reconfigurable computing would allow for efficient customization.

References

- [1] HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. System architecture directions for network sensors. In *ASPLOS* (Cambridge, November 2000).