A System Architecture for Networked Sensors

Jason Hill, Robert Szewczyk, Alec Woo,
Seth Hollar, David Culler, Kris Pister

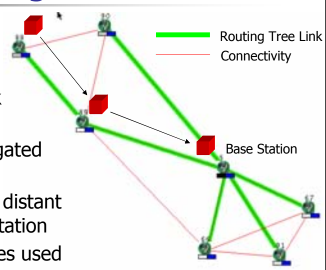
http://tinyos.millennium.berkeley.edu
U.C. Berkeley
11/13/2000

Computing in a cubic millimeter:

- Advances in low power wireless communication technology and micro-electromechanical sensors (MEMS) transducers make this possible
- How do you combine sensing, communication and computation into a complete architecture
- What are the requirements of the software?
- How do you evaluate a given design?

Ad hoc sensing

- Autonomous nodes self assembling into a network of sensors
- Sensor information propagated to central collection point
- Intermediate nodes assist distant nodes to reach the base station
- Connectivity and error rates used to infer distance



Organization

- The Vision
- Hardware of today
- Software Requirements
- TinyOS system architecture
- System evaluation

Today's Hardware

1.5" x 1.5"

- Assembled from off-the shelf components
- 4Mhz, 8bit MCU (ATMEL)
  - 512 bytes RAM, 8K ROM
- 900Mhz Radio (RF Monolithics)
  - 10-100 ft. range
- Temperature Sensor & Light Sensor
- LED outputs
- Serial Port

No dedicated I/O controllers

- Bit by bit interaction with Radio
- Software must process bit every 100µs
- No buffering
  - missed deadline → lost data
- Must time share on granularity of 2x sampling rate

Key Software Requirements

- Capable of fine grained concurrency
- Small physical size
- Efficient Resource Utilization
- Highly Modular

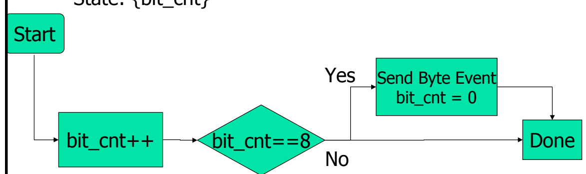
TinyOS system architecture

State Machine Programming Model

- System composed of state machines
- Command and event handlers transition a module from one state to another
  - Quick, low overhead, non-blocking state transitions
- Many independent modules allowed to efficiently share a single execution context

Simple State Machine Logic

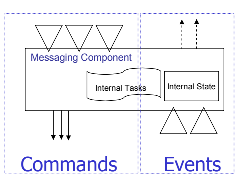
Bit_Arrival_Event_Handler
State: {bit_cnt}



- Don't you have to do computational work eventually?
- "Tasks" used to perform computational work

TinyOS component model

- Component has:
  - Frame (storage)
  - Tasks (computation)
  - Command and Event Interface

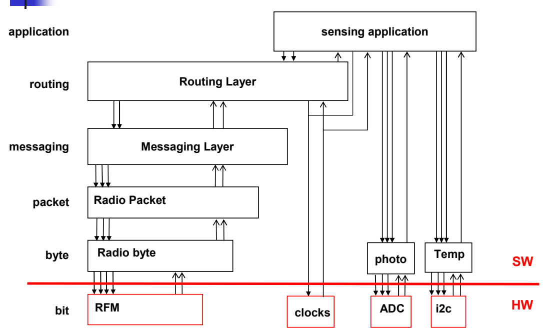


- Constrained Storage Model allows compile time memory allocation
- Provides efficient modularity
- Explicit Interfaces help with robustness

TinyOS – The Software

- Scheduler and graph of components
  - constrained two-level scheduling model: tasks + events
- Provides a component based model abstracting hardware specifics from application programmer
- Capable of maintaining fine grained concurrency
- Can interchange system components to get application specific functionality

Composition into a Complete Application



The Application

- The single node application is just another state machine

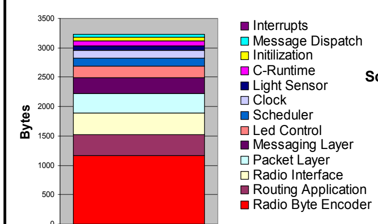
```
Message_Handler(incoming_message){
        if(sender_is_better_parent()){
                my_parent = sender();
        }else if(I_am_parent_of_sender(){
                forward_message(my_parent,
                                incoming_message);
        }
}
Clock_Event_Handler(){
        check_expire(my_parent);
        if(my_parent != null){
                send_data(my_parent);
        }
}
```


Analysis

- Let's take apart Space, Power and Time

Space Breakdown…

Code size for ad hoc networking application



Scheduler: 144 Bytes code
Totals: 3430 Bytes code
226 Bytes data

Power Breakdown…

	Active	Idle	Sleep
CPU	5 mA	2 mA	5 μA
Radio	7 mA (TX)	4.5 mA (RX)	5 μA
EE-Prom	3 mA	0	0
LED's	4 mA	0	0
Photo Diode	200 μA	0	0
Temperature	200 μA	0	0

Panasonic CR2354 560 mAh

- But what does this mean?
  - Lithium Battery runs for 35 hours at peak load and years at minimum load!
    - That's three orders of magnitude difference!
  - A one byte transmission uses the same energy as approx 11000 cycles of computation.

Time Breakdown…

Components	Packet reception work breakdown	CPU Utilization	Energy (nj/Bit)
AM	0.05%	0.20%	0.33
Packet	1.12%	0.51%	7.58
Ratio handler	26.87%	12.16%	182.38
Radio decode thread	5.48%	2.48%	37.2
RFM	66.48%	30.08%	451.17
Radio Reception	-	-	1350
Idle	-	54.75%	-
Total	100.00%	100.00%	2028.66

- 50 cycle thread overhead (6 byte copies)
- 10 cycle event overhead (1.25 byte copies)

How well did we meet the requirements?

- ✓ Capable of fine grained concurrency
- ✓ Small physical size
- ✓ Efficient Resource Utilization
- ✓ Highly Modular

Conclusions

- People are working on shrinking sensors and communication, we need to focus on what brings them together
- TinyOS is a highly modular software environment tailored to the requirements of Network Sensors, stressing efficiency, modularity and concurrency
- We now have a whole new set of tradeoffs need to investigate

Hardware Kits

- Two Board Sandwich
  - Main CPU board with Radio Communication
  - Secondary Sensor Board
- Allows for expansion and customization
- Current sensors include: Acceleration, Magnetic Field, Temperature, Pressure, Humidity, Light, and RF Signal Strength
- Can control RF transmission strength & Sense Reception Strength

How to get more information:

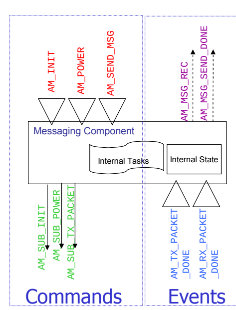
http://tinyos.millennium.berkeley.edu

Real time operating systems

Name	Code Size	Target CPU
pOSEK	2K	Microcontrollers
pSOSystem		PII->ARM Thumb
VxWorks	286K	Pentium -> Strong ARM
QNX Nutrino	>100K	Pentium II -> NEC
QNX RealTime	100K	Pentium II -> SH4
OS-9		Pentium -> SH4
Chorus OS	10K	Pentium -> Strong ARM
ARIEL	19K	SH2, ARM Thumb
Creem	560 bytes	ATMEL 8051

- QNX context switch = 2400 cycles on x86
- pOSEK context switch > 40 µs
- Creem -> no preemption

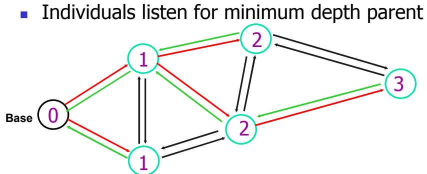
TOS Component




```
                  //AM.comp//
TOS_MODULE AM;
ACCEPTS{
  char AM_SEND_MSG(char addr, char
   type,
                 char* data);
  void AM_POWER(char mode);
  char AM_INIT();
};
SIGNALS{
  char AM_MSG_REC(char type,
                char* data);
  char AM_MSG_SEND_DONE(char success);
};
HANDLES{
  char AM_TX_PACKET_DONE(char success);
  char AM_RX_PACKET_DONE(char* packet);
};
USES{
  char AM_SUB_TX_PACKET(char* data);
  void AM_SUB_POWER(char mode);
  char AM_SUB_INIT();
};
```

Ad hoc networking

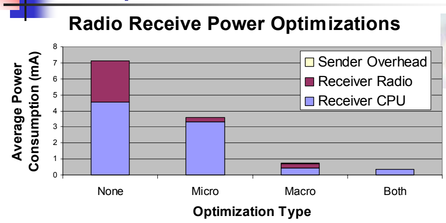
- Each node needs to determine it's parent and its depth in the tree
- Each node broadcasts out <identity, depth, data> when parent is known
- At start, Base Station knows it is at depth 0
  - It send out <Base ID, 0, **>
- Individuals listen for minimum depth parent



Easy Migration of the Hardware Software Boundary

- TinyOS component models hardware abstractions in software
- Component model allows migration of software components into hardware
- Example:
  - Bit level radio processing component could be implemented as specialized FIFO with complex pattern matching
  - Could reduce CPU utilization while sending by more than 50%

Sample tradeoffs



Panasonic CR2354 560 mAh

Battery Lifetime for sensor reporting every minute		
	Duty Cycle	Estimated Battery Life
Full Time Listen	100%	3 Days
Full Time Low_Power Listen	100%	6.54 Days
Periodic Multi-Hop Listening	10%	65 Days
No Listen (no Multi-hop)	0.01%	Years